Final Report
Automatic Control, Project Course
EL2421

Daniel Bug
bug@kth.se

Ioannis Chatzis
ioannisc@kth.se

Anders Cioran
cioran@kth.se

Markus Fjällid
fjallid@kth.se

Linus Flodin
linusflo@kth.se

Anton Hou
ahou@kth.se

Kevin Kyeong
taihyun@kth.se

Marcus Lind
marculin@kth.se

Simon Lundberg
silu@kth.se

Lionel Nurweze
nurweze@kth.se

January 17, 2014

Abstract

In this project, a hardware-in-the-loop (HIL) platform for developing and testing traffic control algorithms was established.

The platform consists of remote control (RC) miniature trucks, virtual models of the miniature trucks, a virtual model of a Scania truck. All these truck models are 1:14 scaled down versions of an actual Scania truck, and the purpose of the HIL is to create traffic scenarios with truck models which are as realistic as possible. All models interact with each other in real-time simulation.

Using vehicle-to-vehicle (V2V) communication in which vehicles share information about themselves and cooperate with each other, traffic control algorithms were developed under a few scenarios. The main scenarios were a four-way crossing and a highway entrance ramp. Other slightly less complex scenarios were also demonstrated. The platform is scalable and can be further improved and extended with more scenarios and complexities to study traffic optimization algorithms and cooperation.

The project included ten Master's students and took place over the course of eight weeks in the Autumn of 2013 at KTH Royal Institute of Technology in Stockholm, Sweden.

Contents

1 Introduction **7**
- 1.1 Background . . . 7
- 1.2 Course Objectives . . . 7
- 1.3 Project Description . . . 7
  - 1.3.1 Description . . . 7
  - 1.3.2 Summary . . . 8

2 Systems Integration **9**
- 2.1 Motion Capture . . . 10
- 2.2 Miniature Trucks . . . 10
  - 2.2.1 TMote Sky . . . 11
- 2.3 xPC Target . . . 11
  - 2.3.1 Scope . . . 11
  - 2.3.2 Communications . . . 11
  - 2.3.3 Real-time Execution . . . 13
- 2.4 Main Program . . . 13
  - 2.4.1 Front panel . . . 13
  - 2.4.2 Program structure . . . 13
- 2.5 Integration . . . 14
- 2.6 Visualization . . . 15
  - 2.6.1 Step 1 . . . 15
  - 2.6.2 Step 2 . . . 16
  - 2.6.3 Step 3 . . . 16

3 Map Creation **19**
- 3.1 Roads . . . 19
  - 3.1.1 Signs and Traffic Lights . . . 19
  - 3.1.2 Transitions . . . 19
  - 3.1.3 Road data . . . 20
- 3.2 Crossing Scenario with Transition Points . . . 20
  - 3.2.1 Function . . . 20
  - 3.2.2 Create transitions . . . 21
  - 3.2.3 How right and left are detected . . . 21
  - 3.2.4 How the *CrossingMatrix* is used . . . 21
- 3.3 Highway Scenarios with a Ramp . . . 22

4 Models **24**
4.1 Miniature truck models . . . 24
4.1.1 Velocity dynamics block . . . 24
4.1.2 Steering signal block . . . 25
4.1.3 Vehicle kinematics block . . . 26
4.2 Real truck model . . . 26
4.2.1 Drivetrain . . . 27
4.2.2 Control unit . . . 30
4.2.3 Velocity controller . . . 34
4.2.4 Steering kinematics . . . 35
4.2.5 Limitations . . . 35

5 Low Level Control **36**
5.1 Velocity control . . . 36
5.2 Trajectory Following . . . 36

6 High Level Control **39**
6.1 Centralized and Decentralized Traffic Control . . . 39
6.1.1 Traffic Rules Hierarchy . . . 39
6.1.2 Traffic Lights and Traffic Signs . . . 39
6.1.3 Traffic Lights . . . 40
6.1.4 Traffic Signs . . . 41
6.1.5 Right-hand Rule . . . 41
6.2 Collision Avoidance . . . 42
6.3 Vehicle-To-Vehicle Communication . . . 44
6.4 Crossing Scenario . . . 44
6.4.1 Information exchange . . . 44
6.4.2 Algorithm . . . 45
6.5 Highway Entrance Scenario . . . 46
6.5.1 Information exchanges . . . 47
6.5.2 Algorithm . . . 47
6.5.3 Problems . . . 48

7 Conclusions **50**

8 Suggestions for future work **51**
8.1 Models and Control . . . 51
8.1.1 Models . . . 51
8.1.2 Control . . . 51
8.2 Traffic Control . . . 52
8.2.1 Crossings . . . 52
8.2.2 Highway Entrance . . . 52
8.3 System integration . . . 52
8.3.1 Visualization . . . 52
8.3.2 xPC Target . . . 52

A Purchase Order **55**

B Scania GCDC CAN Messages **61**

C Data Packets **63**
C.1 Main Program to Visualization tool 63

D Real truck model data **66**

List of Figures

2.1 SML Environment. 9
2.2 Miniature RC Truck of a Scania Model. 10
2.3 xPC Target Platforms 12
2.4 Frontpanel of the main program 13
2.5 Flowchart of the main program 14
2.6 Diagram over system interface 17
2.7 Graphical userinterface of the visualization tool. 18

3.1 Map example with waypoints. 20
3.2 Crossing map example. 22
3.3 Highway entrance example. 23

4.1 The miniature truck model. 24
4.2 Velocity dynamics block. 25
4.3 Steering block structure. 25
4.4 Bicycle kinematic model. 27
4.5 Drivetrain model. 28
4.6 Available engine torque. 29
4.7 Shift decision logic. 32
4.8 Velocity controller (Real truck model). 34

5.1 Trajectory following. 37
5.2 FIR steering controller (de)activation. 38

6.1 Sorted crossing and placement of traffic lights/signs 40
6.2 Alternating of traffic lights in a crossing 40
6.3 Illustration of the collision controller 42
6.4 A standard ellipse, with the radii a and b. 43
6.5 Sections in crossing. 44
6.6 Highway entrance scenario. 46
6.7 General communication setup. 47
6.8 General decision algorithm. 48

B.1 Simulink Block Sets for GCDC20 CAN Message 61
B.2 Simulink Block Sets for GCDC00-07 CAN Messages 62

List of Tables

3.1 Contents of the file *SignInfo.txt* 21
3.2 CrossingMatrix Column Information. 21

4.1 Gear state description I. 31
4.2 Gear state description II. 31
4.3 Gear state update. 32
4.4 Gear shifting procedure. 33

6.1 Shared collision matrix. 45
6.2 Crossing example. 45

D.1 Scania model parameters. 66
D.2 Scania model gear ratios. 67

Chapter 1

Introduction

1.1 Background

Cooperation between vehicles can enable safer, greener and more efficient traffic and transports. Vehicles can communicate their states, such as velocity and position, and hence adapt to the surrounding traffic in order to avoid collision and optimize traffic in crossings, for instance. This adaptation can be done autonomously for a large number of vehicles which will give us a more efficient traffic.

To develop this system of cooperation, a simulation platform needs to be built in order to evaluate the cooperation. Such a platform and cooperation will be described in this report.

1.2 Course Objectives

The course *EL2421 Automatic control, project course* is a 15-credit course running over ten weeks. In the course of 2013, ten Master's students with various engineering disciplines are taking the course. The course aims to teach the students to work in projects and give practical knowledge about modelling and design of control systems.

The course takes place in the *Smart Mobility Lab (SML)* at the *Royal Institute of Technology (KTH)* in Stockholm, Sweden. A new project is created for every year's course and the result is supposed to contribute to the lab.

1.3 Project Description

This section will give a short description of the project goals. To access the *purchased order* given to the group at the beginning of the project, please see Appendix A

1.3.1 Description

The project aims to develop a hardware-in-the-loop simulation platform including both radio controlled real miniature trucks and simulated models of trucks.

Trucks

The SML possesses radio controlled miniature trucks. For this simulation, truck models should be developed for a LabVIEW environment as well as external real time hardware (xPC). Steering and velocity controllers should be built for all trucks.

All types of trucks should be able to drive on the same road network at the same time and in the end be able to interact which each other. The miniature truck models should have the same behavior as the real miniature trucks and enable to more easily simulate complex scenarios without risk of damages due to collisions with real trucks. The full scale Scania truck models behavior correspond to a real truck and can for example help to get a more reality-like scenario.

Control Complexities and Scenarios

Several simulation with different scenarios and different levels of traffic control will be developed. The most advanced scenario include vehicle-to-vehicle communication where trucks cooperate, share information and negotiate on what action they will take. This cooperation needs to be developed. To test the cooperation it will be implemented in two scenarios: a four way crossing and a highway entrance ramp.

There will be three levels of traffic control complexities implemented (taken from the purchased order, see Appendix A): 1) Centralized traffic control, using traffic lights; 2) Decentralized controlled based on traffic rules and pre-defined priorities, for example the right-hand rule; 3) Cooperative decisions where vehicles agree on their actions, no predefined priorities or rules.

1.3.2 Summary

The following is a summary of the order given to the group:

- Design kinematic/dynamic models of the miniature trucks and run them in LabVIEW
- Design a model of a full scale truck and run it on a dedicated xPC target
- Run simulations in real time
- Create an integrated hardware-in-the-loop simulation environment
- Create vehicle controllers for steering and velocity
- Create communication between vehicles
- Create traffic controllers with different levels of complexities
- Implement traffic control in different scenarios
- Visualize the simulation

Chapter 2

Systems Integration

Prior to getting into the details of truck modeling and traffic control scenarios, this chapter introduces different systems installed in the SML, and presents their functionalities and setups. The integration of the systems and their purposes in real world environment are also discussed.

As shown in Figure 2.1, the major components in the SML are motion capture cameras, miniature RC trucks, xPC Target machines, main program in LabVIEW, visualization tool in MATLAB, and ceiling mounted projectors.

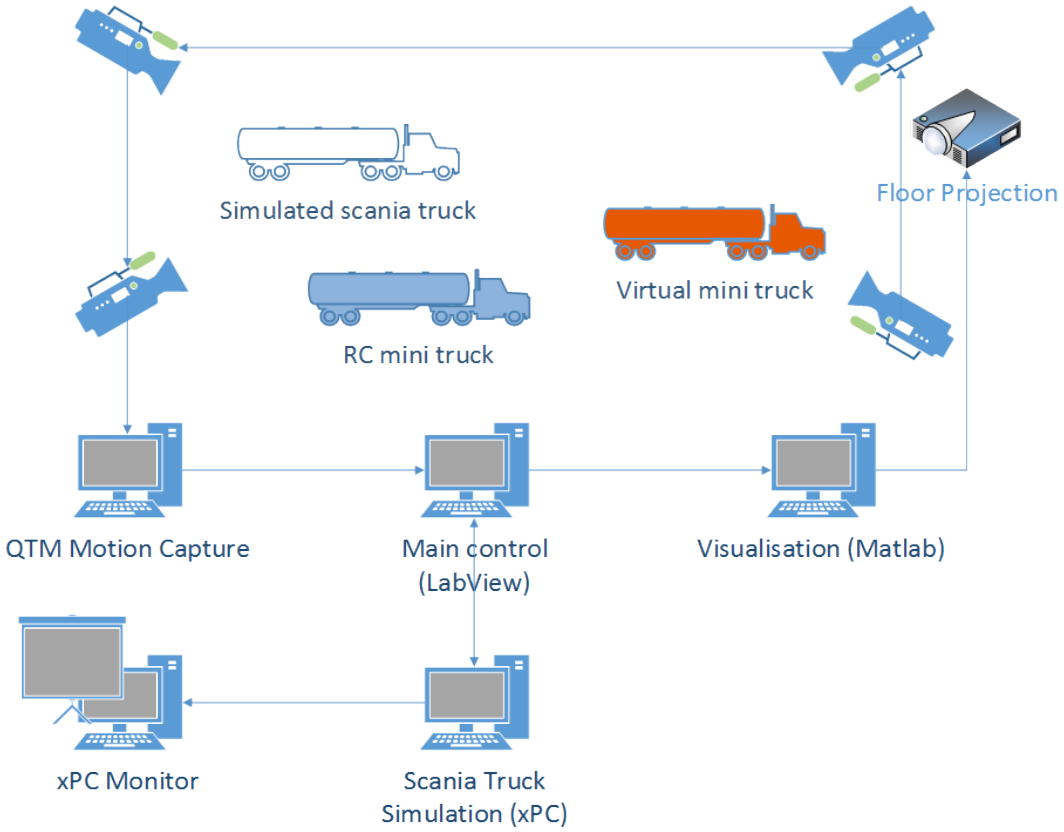


Figure 2.1: SML Environment.

The motion capture cameras are from Qualysis and are handled by Qualisys Tracking Manager (QTM). They are used to track the motion of the miniature RC trucks in the laboratory environment.

The xPC Target machines are real-time systems that run a real life truck model and its self-driving controller. When it comes to the study of the performance of a truck in various traffic scenarios, this is perhaps the core part of the whole platform.

Then, to create more interesting traffic flow, the miniature trucks are modelled in equations and virtualized in the main LabVIEW program. It is also this LabVIEW program where various traffic controls are implemented and all communications between the systems are handled to establish a hardware-in-the-loop (HIL) platform. If the purpose of xPC Target is to study the characteristics of a real truck, the main LabVIEW program is used to study and experiment different traffic control algorithms.

Finally, the visualization tool in MATLAB is used to project road networks, virtual mini-trucks and simulated real life trucks on the floor of the SML for demonstration. Furthermore, a secondary projector serves as the dashboard of the simulated real truck model.

2.1 Motion Capture

The SML is equipped with 12 motion capture cameras placed on the ceiling in such a way that the area where objects are to be tracked is well covered by the cameras. The motion capture system then tracks the motion of objects in this designated area by using infrared reflective markers placed on the objects. Each object has the markers on them with a unique spatial pattern, and such is defined and stored in the QTM. In the scope of this project, the motion capture system has mainly been used for positioning feedback of RC trucks that are included in traffic scenario demonstrations. The detailed guides on setting up and using this system have previously been worked on by the SML technicians Mani, Pedro and Joao, and can be referred to their manuals for more information[2][5].

2.2 Miniature Trucks

In the lab, there are several remote control (RC) trucks which are 1:14 scaled down versions of a Scania truck like the one shown in Figure 2.2. These are battery operated and TMote Sky is installed on each of these miniature truck instead of the originally provided RC device.

Figure 2.2: Miniature RC Truck of a Scania Model.

2.2.1 TMote Sky

TMote Sky is a wireless sensor platform for low-power, high data-rate sensor network applications. It consists of two electric circuit boards: one is placed on either a moving object or a remote station, and the other is usually attached to the central processing station where information from multiple TMote Sky circuits can be collected and shared. The details about the use of TMote Sky and its software TinyOS have also been documented by the aforementioned SML technicians and can be referred to their manuals for more information[2][5].

2.3 xPC Target

xPC Target is a product from MathWorks, and basically is a real-time software environment that allows one to run Simulink and Stateflow models on any x86-based real-time systems. Without having to build or test on a real physical model, a system of interest can be modeled, simulated and studied using xPC Target.

A Scania truck model is thus simulated on an xPC Target hardware platform and controller algorithms are implemented and tested on this virtual model instead of a real truck.

2.3.1 Scope

In the Automatic Control department of KTH, there had been some work done on xPC Target as part of Stockholm Cooperative Driving Project (Scoop). In the Scoop, however, xPC Target was used only to develop a self-driving controller[3]. The scope of the EL2421 project extends this to have an xPC Target platform that represents a real life truck model as accurately as possible. The benefit of this suggested platform is that as the model should well represent an actual truck, it can give out all kinds of physical characteristics of the truck under different traffic scenarios. One of the important and useful characteristics that can be obtained is fuel efficiency with a particular control algorithm applied on the truck platform.

2.3.2 Communications

In this project, two xPC Target platforms have been used: one to virtualize a Scania truck model, and the other to rapid prototype a self-driving controller for the Scania truck. These two systems communicate with each other via controller area network (CAN) and this configuration is completely identical to the connection between a Scania truck and its main electronic control unit (ECU). Thus, xPC Target allows one to fully simulate a system including small details like communication protocols between its subsystems.

The CAN message protocol used in this project has been identical to that of Grand Cooperative Driving Challenge (GCDC) in which the Automatic Control department of KTH participated in cooperation with Scania. For the details of the messages, see Appendix B.

CAN

There are 9 GCDC CAN messages, namely GCDC00, 01, 02, 03, 04, 05, 06, 07, 20. Among them, GCDC20 gets sent to the truck from the controller, and the rest is received

Figure 2.3: xPC Target Platforms

by the controller from the truck.

In GCDC20, in replacement of ExternalAccelerationDemand, a new variable called SteeringAngleReference was defined.

The xPC Target systems are equipped with CAN I/O functionalities and a CAN bus between systems is established by using Simulink block instances. For the xPC Target platform on a custom PC, the CAN I/O board is Softing CAN-AC2-PCI, and corresponding CAN Simulink block instances are included in the installation of Simulink's xPC Target toolbox. On the other hand, the Speedgoat machine requires an additional installation of its own xPC Target toolbox.

When establishing a CAN bus, it is also critical to define what each message contains and how the data in each message is organized. Using provided CAN Simulink block sets, the message format can be manually edited. Nonetheless, there is a more convenient way of doing this, which is creating a single CAN database (*.dbc) and having each CAN node to refer to this database. This way, any future modification to the protocol can be made more readily and any sort of inconsistency between CAN block sets can be minimized. The CAN database was created using CANdb++ editor by Vector.

To ease the learning curve and development time in any future projects on xPC Target, simple examples and debugging tools have been created (see the file *HelloCAN.slx*).

UDP

Besides CAN, one of other communication protocols available on xPC Target is user datagram protocol (UDP). The internet protocol TCP is not supported by xPC Target as it does not have the nature of real-time that UDP does.

This communication protocol was used to simulate the built-in wireless communication system in Scania trucks. In the lab environment, this was simply to communicate between xPC and the main LabVIEW program.

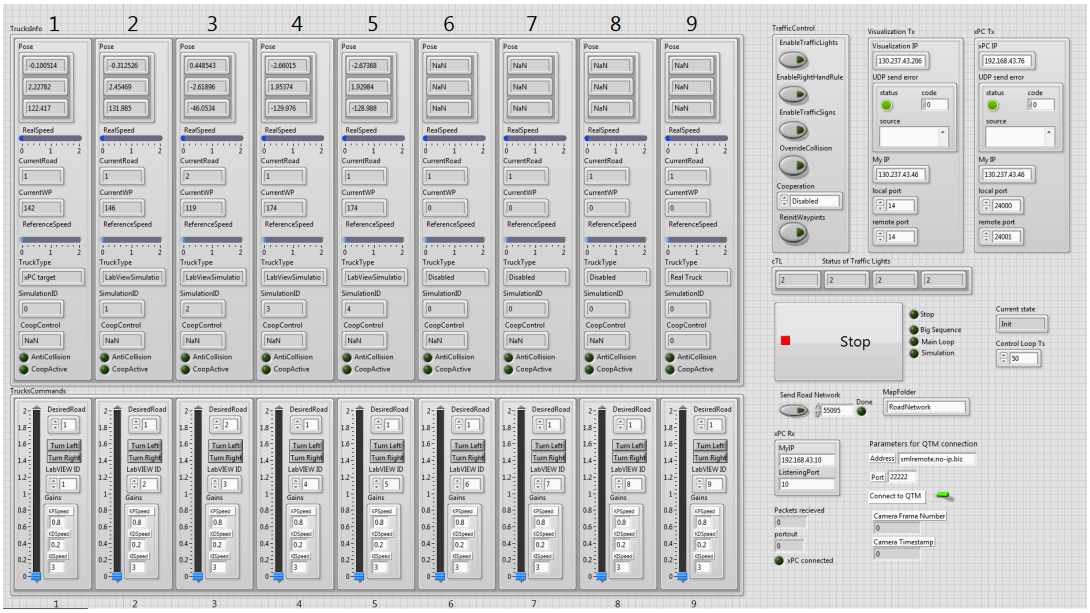


Figure 2.4: Frontpanel of the main program

To ease the learning curve and development time in any future projects on xPC Target, simple examples and debugging tools have been created (see the file *HelloUDP.slx*).

2.3.3 Real-time Execution

In order for a Simulink/Stateflow model to be programmed onto xPC Target and be properly run in real-time, the model needs to have certain configuration settings correctly. Most importantly, the chosen solver must be discrete-time. Furthermore, the selected sample time should not be any less than the average task execution time (TET) of the model. The recommended sample time is at least ten times greater than the maximum TET of the model.

2.4 Main Program

The main program is created in LabVIEW. It is based on a pre-existing program written as part of the Master's theses by P. Lima[6] and J. Alvito[1]. A lot of basic functionalities have been be reused, i.e. controllers for trajectory following and connections to TMote Sky and QTM.

2.4.1 Front panel

The front panel seen in Figure 2.4 serves as the main user interface when the program is running. It features a lot of buttons, sliders, and indicators to control and monitor the operation.

2.4.2 Program structure

The structure of the program is displayed in Figure 2.5.

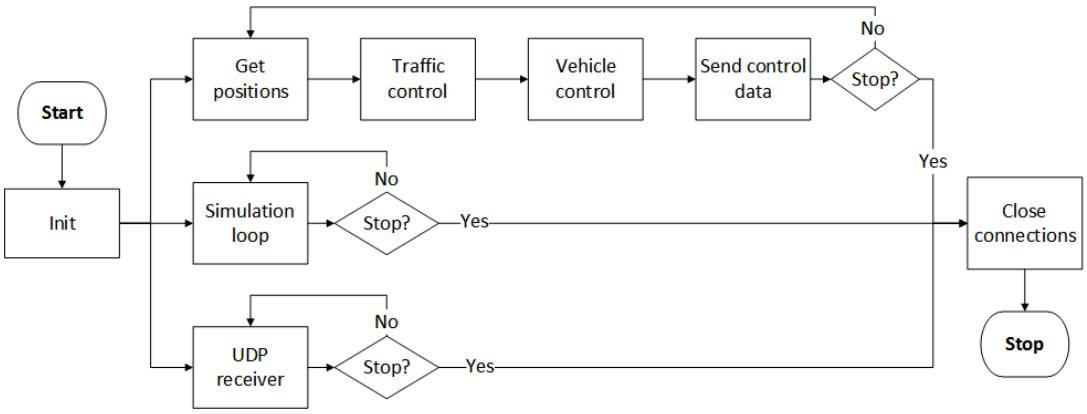


Figure 2.5: Flowchart of the main program

When the program is started, the following initialization phase takes place.

- Variables are initiated
- Road network files are read in to the memory
- Initial positions for the simulate vehicles are calculated

The three following branches in the flow represent processes running in parallel i.e. asynchronous. The uppermost branch is what is referred to as the main loop.

Get positions updates x,y and yaw position variables for each active vehicle. Positions for real Mini Trucks are collected from the motion capture system, positions for simulated Mini Trucks are collected from the parallel process *Simulation loop* and positions for the simulated Scania Truck are collected from the XPc target machine through the *UDP receiver* process.

Traffic control process runs different traffic control algorithms depending on settings in the front panel. The algorithm runs in a for-loop that iterates once for each vehicle. The inputs are vehicle position, desired speed from front panel, road network etc. The outputs are a reference speed and waypoint.

The *Vehicle control* process uses the outputs from *Traffic control* as references for speed and steering controllers.

In *Send control data* the control values from *Vehicle control* are transferred to corresponding vehicle.

The main loop is also part of the actual control loop. The sampling time T_s is set by altering a time delay in the *Send control data* process.

2.5 Integration

The separate devices in the system are connected using internet protocols (IPs). An overview of the setup can be seen in Figure 2.6 and packet structures are listed in Appendix C.

The motion capture system is connected to the Main Program using TCP. In the Main Program the connection is handled within a closed LabVIEW library provided by Qualisys.

Problems were detected when a LabVIEW program receives UDP packets from the xPC. For an unknown reason the reception rated dropped to about one packet in five

seconds. A simple port forwarder is implemented as a workaround for this issue. The port forwarder is written in Python and is running on the same computer as the LabVIEW program, i.e. Main Program and xPC monitor. It receives a packet on specified UDP port and forwards it to another UDP port on the localhost.

2.6 Visualization

The visualization tool is entirely written in MATLAB. It receives the map to project from the LabVIEW computer using TCP/IP transmission and the data used for displaying the trucks are received using UDP. A local copy of the map can also be loaded but it is important to always use the same map as LabVIEW. The code for the visualization tool is based on Jonathan and Eriks summer project [7].

The graphical user interface is divided in three logical steps as shown in Figure 2.7. All the different buttons and inputs of the interface are described in the proceeding sections. When the projection button in the third step is pressed the graphical map is generated. From the packets received from LabVIEW overlaying graphics is generated. This includes traffic lights, trucks and informative texts. When a new UDP package is received from LabVIEW the overlaid graphics representing the trucks etc. are updated to the newest positions and states. The framerate is decided by the speed of which LabVIEW is sending the UDP packages. The maximum framerate is however limited by how fast MATLAB can update the graphical objects. Experiments have shown that around 30 frames per second is maximum in our setup.

2.6.1 Step 1

In the first step general settings are applied regarding appearance and network connection.

Show waypoints Display arrows on all the waypoints if ticked.

Show Bridge Not used.

Show static map (without trucks) If ticked only the map will be displayed. No trucks are shown.

Hide physical trucks Hides the blue box for the physical trucks if ticked.

Add marker in origo Shows a cross in origo if ticked. Useful for calibration and centering.

Run on same PC as LabVIEW If ticked the program will listen for data on the localhost.

Jonas mode All trucks are replaced with images of Jonas.

Color theme Select a color theme for the projection.

IP Host Computer IP address of the computer transmitting the map and sending truck data.

UDP receive port This specifies on which port the truck data is received on.

2.6.2 Step 2

In the second step one reads the map either from a local copy or received from LabVIEW.

Load local map Reads a local copy of a map saved on the computer.

Host Computer port Specifies which port to use for the TCP/IP transmission.

Transfer map via TCP/IP Click this button to receive an pending map transmission from LabVIEW.

Map received Indicates that all files needed for the map has been received.

2.6.3 Step 3

In the third step the actual projection is started by pressing the huge button. A new figure appears on the secondary monitor.

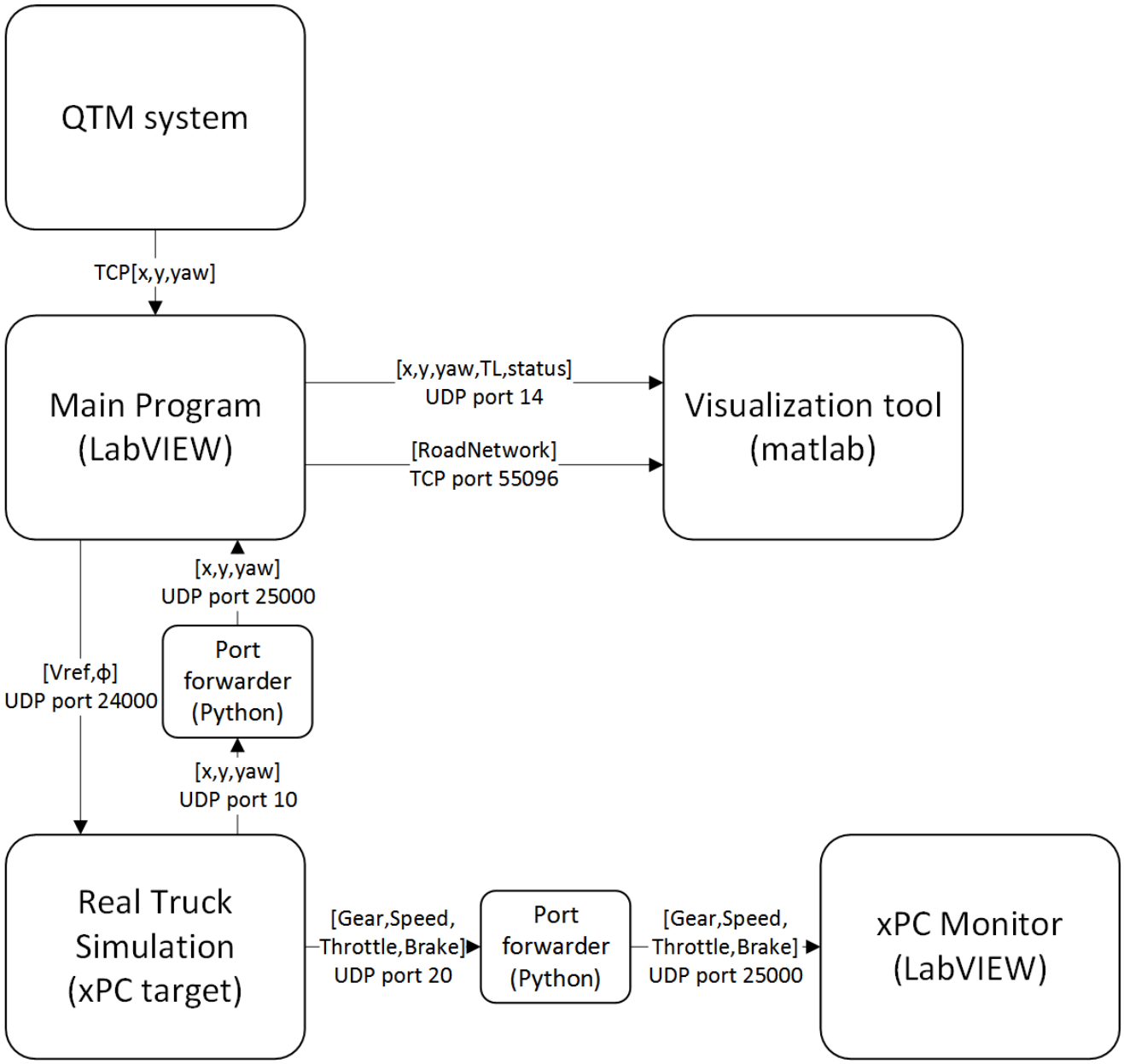


Figure 2.6: Diagram over system interface

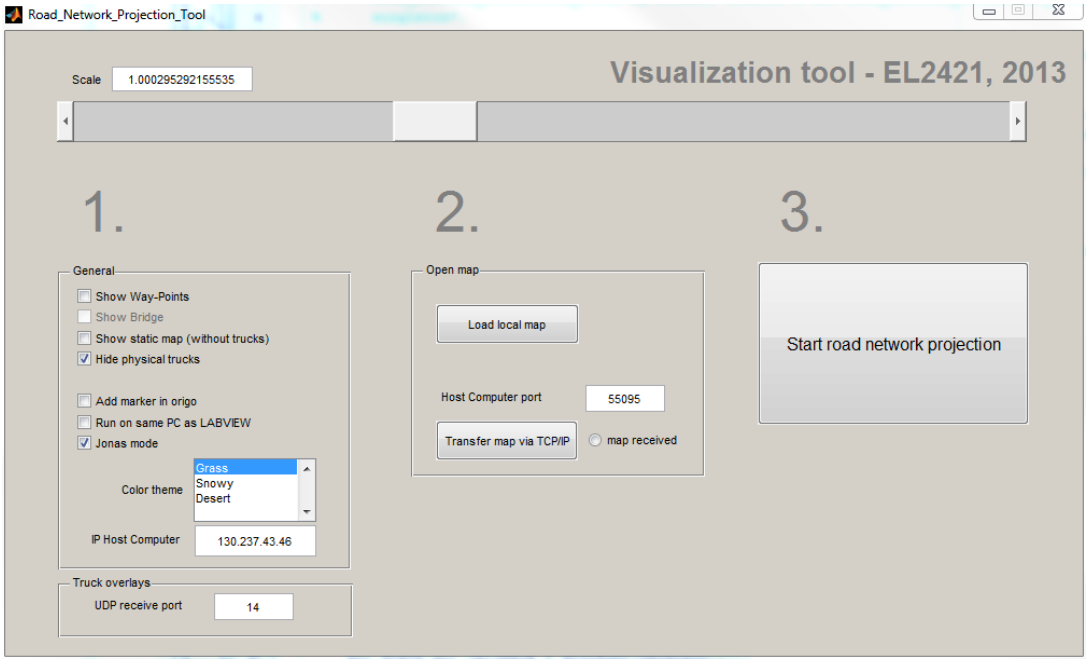


Figure 2.7: Graphical userinterface of the visualization tool.

Chapter 3

Map Creation

One advantage with a good simulation platform is the ability to rapidly change scenarios and traffic situations. The map creation tool, *Road_Network_Creator*, inherited from the previous project, has been used [7]. This chapter will describe how roads are created for different scenarios and what is important to take into account. It will also give a brief explanation of files and information associated with the road network that the simulation platform uses.

3.1 Roads

The road is represented by waypoints which are interpolated between manually placed marks in the *Road_Network_Creator*. Each waypoint has a number, starting from zero, and an associated x- and y-coordinate. Figure 3.1 shows an example where every point represents a waypoint.

When creating a road network, several files are created of which some are explained here.

3.1.1 Signs and Traffic Lights

Several traffic signs are implemented and can be used. Signs are associated with a waypoint which, if necessary, can be changed manually in the file *SignInfo.txt*. Every column in this file represents a sign and includes the information in Table 3.1

Traffic light information is stored in the file *TLInfo.txt* and includes information about road number, waypoint and coordinates [7].

3.1.2 Transitions

Transitions are a way to change road, turn or make a shortcut between waypoints. Each transition starts with a *FromState* and ends with a *ToState*. Transition point data are stored in the file *TransitionMatrix.txt*. Every transition has a number represented by the column, line one and two tell us which road the *FromState* is and its waypoint number. Line three and four give the same information but about the *ToState*.

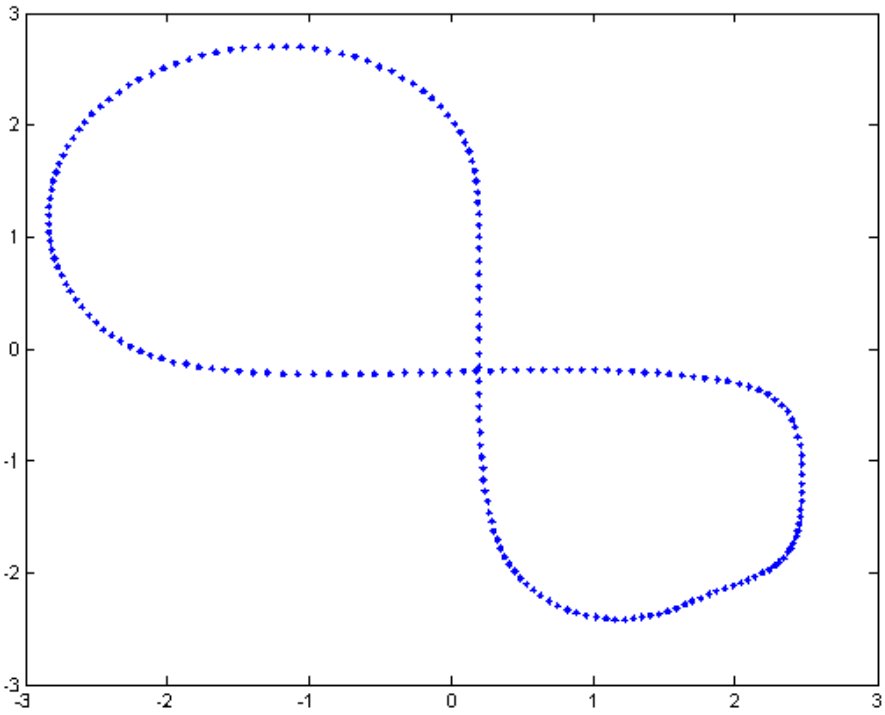


Figure 3.1: Map example with waypoints. Each point represents a waypoint used with path finding.

3.1.3 Road data

A lot of data about the road are stored as well. The file *savedLUT.txt* contains distances (in $[m]$) between every waypoints on the road. Note that the distance is along the road.

The file *savedRoads.txt* contains X- and Y-coordinates for every waypoint (column) on every road where X and Y for road one is on line one and two, for road two on line three and four and so on [7].

3.2 Crossing Scenario with Transition Points

3.2.1 Function

To be able to use any crossing function such as the right-hand rule, crossing cooperation, and to turn right and left, vehicles must know orientations inside a crossing. When a map is read by the LabVIEW program, the code will attempt to find a valid crossing. If a crossing is detected, a variable matrix is created, the *CrossingMatrix*.

The *CrossingMatrix* consists of all the information needed for a truck to quickly orientate inside a crossing. Each column contains information for one input-lane, where the columns are sorted in a counter-clockwise fashion per input-lane. The information found in the *CrossingMatrix* is according to Table 3.2.

Table 3.1: Contents of the file *SignInfo.txt*

Information	Notes
Road	
Waypoint	
X-coordinate	
Y-coordinate	
Type of sign	1=Stop, 2=Yield, 3=Velocity, 4=Pedestrian crossing
Velocity	Only for velocity signs

Table 3.2: CrossingMatrix Column Information.

Row number	Column information
1	FromState Road number
2	FromState Way Point number
3	ToState Road number (Transition point row 1)
4	ToState Way Point number (Transition point row 1)
5	Next Road number if RIGHT Turn
6	Next Way Point number if RIGHT Turn
7	Next Road number if LEFT Turn
8	Next Way Point number if LEFT Turn

3.2.2 Create transitions

When creating a four-way crossing, one needs to place exactly four transition points in the crossing area. The four transition points must be placed in such a way that all states seen in Figure 3.2 are covered, where the from-states and to-states match. Note that there should be no transition point beginning or ending inside the crossing.

The road that was used for testing had the shape of an infinity-loop with a two-way single-lane road. However, any road configuration could work, as long as the transition points are set correctly.

The program is coded to detect four transition points in vicinity to each other, which is the main condition for it to detect a crossing.

3.2.3 How right and left are detected

If transition points are placed correctly, the program will be able to create the *CrossingMatrix*. Every entrance (*FromState*) to the crossing is represented by a column and every column contains information about what road and waypoint (*ToState*) the vehicle has on its right and left.

The program compares the X- and Y-coordinate of the *ToState*s and by finding the extreme values it is able to sort them in an anti-clockwise direction. From this the *ToState*s, to the right and left are sorted into the *CrossingMatrix*.

3.2.4 How the *CrossingMatrix* is used

When a vehicle is approaching a crossing and intends to follow a traffic rule or turn it checks the *CrossingMatrix* for a column that corresponds with its own position (road and

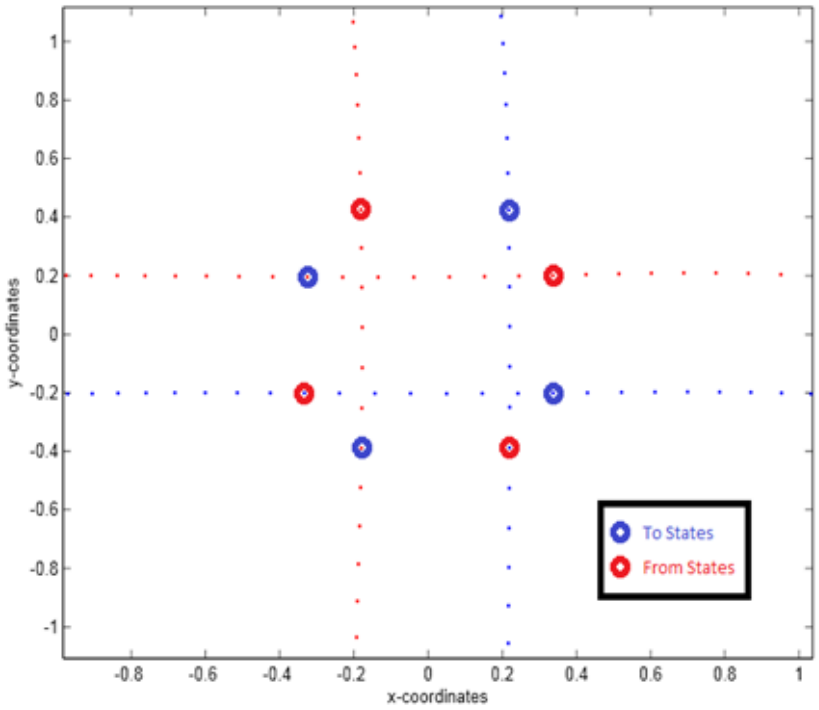


Figure 3.2: A crossing map created in the map-creator with indicated points for the four transition points. Each line represents a road.

waypoint). When it finds it, it can easily detect what waypoints is on its right and left side.

If a transition point is located at a waypoint before a traffic light or sign and the vehicle intends to turn it will not react to the sign/light. Make sure that the traffic light or sign is located at a waypoint in front of the transition and if necessary manually change the transition point.

3.3 Highway Scenarios with a Ramp

The highway entrance scenario requires at least two roads, a yield sign and a transition point. A highway entrance is detected if and only if there is an yield sign followed (within 50 waypoints) by a transition point from the same road. The transition point will then act as the entrance point from the regular road to the highway road. Therefore, the two roads should preferably be very close to each other (almost overlapping) at the transition point.

Because of how the implemented algorithm works, the transition point should be positioned near the end of a straight road section where the two roads go in parallel.

As an example, the road structure as in Figure 3.3 for testing the scenario was used.

For more realistic performance, it is also suitable to add speed signs wherever necessary. Suggestively, a high speed sign on the highway and one close to the yield sign, and a lower speed sign at the beginning of the regular road.

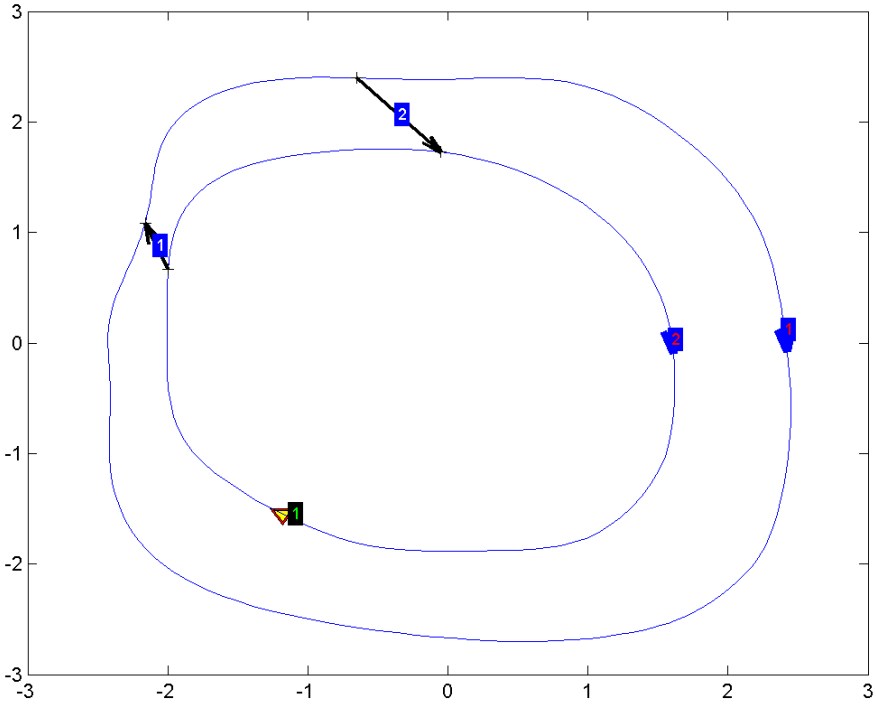


Figure 3.3: Example of a suitable highway entrance map created in *Road_Network_Creator*. Outer road: highway, inner road: regular road. The second transition point is for testing purposes (for setting trucks back on the inner road after having existed to the outer road).

Chapter 4

Models

Different types of vehicles are used in the simulation platform, each with their own advantages and disadvantages. This chapter will describe the two different models used and developed: The miniature truck models and the real truck model.

4.1 Miniature truck models

The miniature truck model captures the simple dynamics of Tamiya RC Scania R620, taking into account some physical limits. No gearing is taken into consideration, since the miniature trucks in the experiments only drive on one gear as well. The miniature truck model block is illustrated in Figure 4.1.

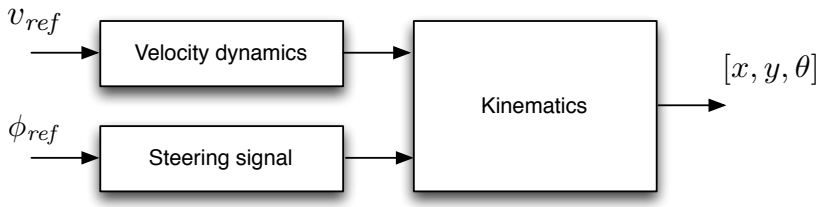


Figure 4.1: The miniature truck model.

The inputs to the model are the velocity, v_{ref} and steering angle, ϕ_{ref} signals, while the outputs are the position of the miniature trucks (x, y) and the yaw angle θ. The miniature truck model consists of three main blocks: *a*) The velocity dynamics block; *b*) the steering signal block; and the *c*) vehicle kinematics block.

4.1.1 Velocity dynamics block

The structure of the velocity dynamics block is portrayed in Figure 4.2. The velocity dynamics block consists of the following three sub-blocks:

- The communication delay block
- The saturation block
- The first-order velocity dynamics block

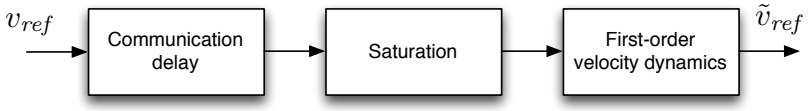


Figure 4.2: The structure of the velocity dynamics block.

Communication delay block

The *communication delay block* models the time required for the input signal v_{ref} to reach the receiver in the miniature truck through the wireless link. The communication delay, t_{delay} used in the project is $t_{\text{delay}} = 0.16$ [s] [4].

Saturation block

The *saturation block* models the minimum and maximum velocity v_{min} and v_{max} respectively, the miniature trucks can achieve. The upper limit of the saturation block which corresponds to the maximum attainable velocity is $v_{\text{max}} = 2$ [m/s] while the lower limit is $v_{\text{min}} = 0$ [m/s], which is corresponding to the lowest attainable velocity [4].

First-order velocity dynamics block

The *first-order velocity dynamics block* models the inertia of the miniature truck which limits the rate of change of the velocity. It is represented by a first-order transfer function $F(s)$ such that:

$$F(s) = \frac{1}{\tau s + 1} \tag{4.1}$$

where τ is the time constant. The time constant used in the project is $\tau = 0.3$ [s] was manually adapted to match the real trucks behavior.

4.1.2 Steering signal block

The structure of the steering signal block is portrayed in Figure 4.3. The steering signal block consists of the following four sub-blocks:

- The communication delay block
- The saturation block
- The rate limiter block
- The backlash block

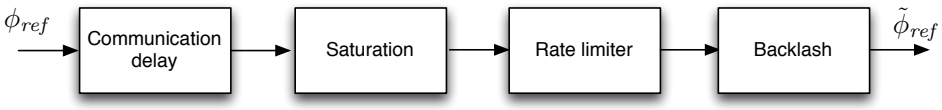


Figure 4.3: The structure of the steering signal block.

Communication delay block

The *communication delay block* serves the same way as the one included in the velocity dynamics block and as a result has the same value $t_{\text{delay}} = 0.16$ [s].

Saturation block

The *saturation block* models the minimum and maximum steering angles, $\phi_{\min}$ and $\phi_{\max}$ respectively, the miniature trucks can achieve. The upper limit of the saturation block which corresponds to the maximum left steering angle is $\phi_{\max} = 0.4$ [rad], while the lower limit is $\phi_{\min} = -0.4$ [rad], corresponding to the maximum achievable right steering angle. These limits where determined by measuring the maximum and minimum steering angle of the miniature trucks.

Rate limiter block

The *rate limiter block* limits the rate of change of the steering angle ϕ. The limits of the rate limiter block are ± 365 [rad/s] [4].

Backlash block

Finally, the *backlash block* models the deadzone before a change from a positive to a negative steering angle is applied and vice versa. The deadband of the backlash block used in the project is $\Delta = 0.02$ [rad] which is the 5% of the maximum steering angle $\phi_{\max}$.

4.1.3 Vehicle kinematics block

The kinematic vehicle model used for the trucks is the *bicycle model*. The bicycle model is illustrated in Figure 4.4. Assuming that the velocity v is constant the bicycle model is described by Equation 4.2 - Equation 4.4.

$$\dot{x}(t) = v(t)\cos(\phi(t) + \theta(t)) \tag{4.2}$$

$$\dot{y}(t) = v(t)\sin(\phi(t) + \theta(t)) \tag{4.3}$$

$$\dot{\theta}(t) = \frac{v(t)}{l}\tan\phi(t), \tag{4.4}$$

where x and y are the Cartesian coordinates of the rear position of the miniature truck, θ and ϕ are the yaw and steering angle respectively, v is the miniature truck velocity, l is the length between the axles of the miniature truck, and $\dot{\theta}$ is the yaw rate.

4.2 Real truck model

In this section, the model of the real truck that is simulated in real time on the xPC is explained. The purpose of the model is to make it possible to see how a real truck would behave in the system. To do this, a model is created in Simulink including the different components of a real truck drivetrain and related control systems.

As a way to obtain realistic values on parameters, Scania provide the project with data that is logged in an actual truck when driving. Some of the parameters in the model

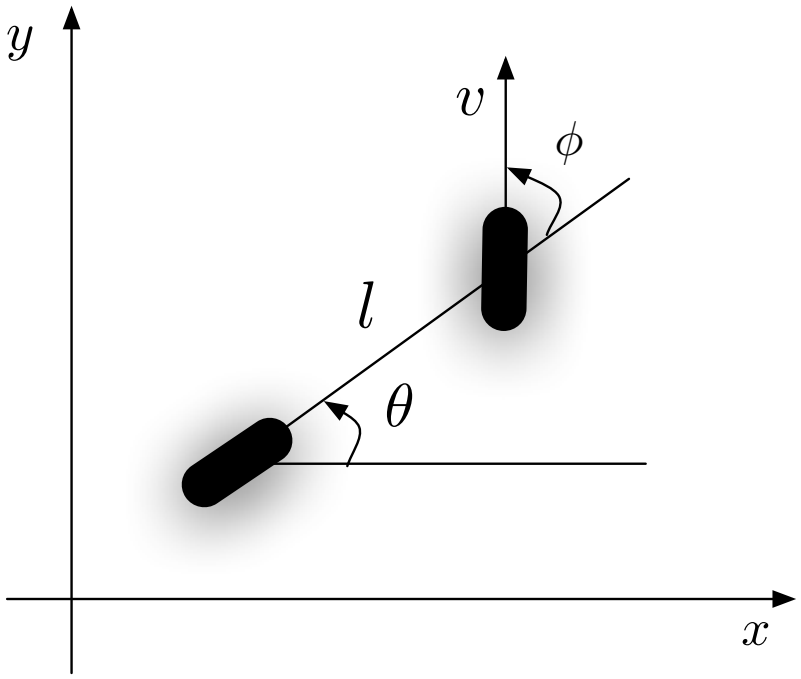


Figure 4.4: The bicycle kinematic model.

can then be identified in this way, which makes it possible to create a better model. Besides this, some parameter values are obtained directly from Scania.

4.2.1 Drivetrain

The drivetrain of a truck can be divided into several main components, see Figure 4.5, including engine, clutch, gearbox and differential. This part of the model captures how the torque from the engine is transferred to the wheels and thus to the vehicle body. The input to this part of the model is the actual throttle level, $\hat{\delta}$ and the brake level, β. These signals are limited so that $\hat{\delta}, \beta \in [0, 1]$, where the upper limit implies full throttle/brake.

The drivetrain is modeled using the physical modeling toolbox, Simscape, within Simulink. This means that the drivetrain is connected in a physical network conserving the power as it is transferred between the components.

In physical modeling one often talks about through and across variables. A through variable is transmitted through an element in the system, for example current that is transferred through one side to the other in a resistor. An across variable is the difference between two points in an element, for example voltage that is the difference between two levels of potential.

Since the product of the variables has the unit of power, they can be related by a set of equations in every element in a way so that the power is always conserved in the physical network.

In the drivetrain model, these variables are torque and angular velocity respectively except for the last part of the drivetrain model which converts the power in to the translational domain with force and translational velocity.

The gearing system is modeled as semi automatic, which is often the case in real trucks. This means that there is a clutch and a gearbox which are controlled automatically by a

control system. The commands from a driver or, as in this case, the velocity controller is therefore an accelerator signal and a brake signal.

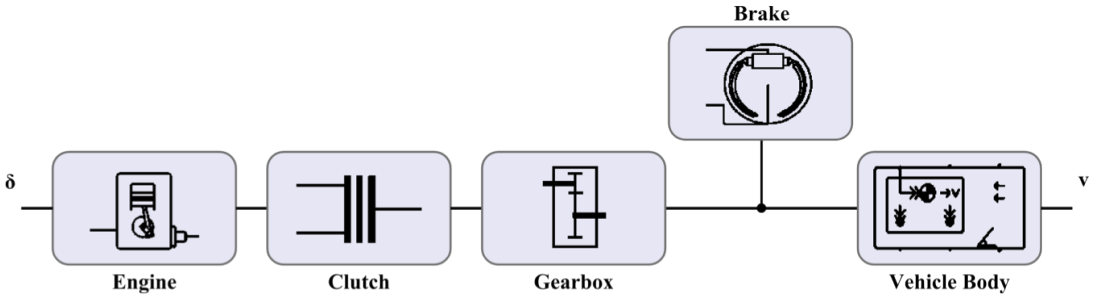


Figure 4.5: Drivetrain model.

Engine

The engine is modeled using an ideal torque source together with a look up table for the available torque. See Figure 4.6 for the data that is used in the model. The look up table provides the available maximum torque from the engine at current engine speed, $T_{\max}(n_{\text{engine}})$, which is then multiplied by the throttle level. Some engine friction, T_{friction} is subtracted to give the actual torque. The engine model also includes an inertia, J_{engine}. The engine dynamics can therefore be described by the following set of equations when standalone.

$$J_{\text{engine}}\dot{\omega}_{\text{engine}} = T_{max}(n_{\text{engine}})\delta - T_{\text{friction}}, \tag{4.5}$$

where ω_{engine} is the angular speed of the engine. The engine speed and actual torque are measured for use by other parts of the system.

Numerical values used in this part of the model are $J_{\text{engine}} = 65.8$ [kg/m^2] [**?**] together with the engine torque data obtained from Scania. T_{friction} is set to 50 [Nm] in order to get a realistic deceleration of engine speed when disconnected from the rest of the drivetrain.

Clutch

The purpose of the clutch is to make it possible to control the torque that is transferred from the engine to the rest of the drivetrain. This enables starting from standstill and decoupling of the engine when shifting gear. This component is modeled using a disk friction clutch component in Simscape, which models how the torque is transferred between two shafts by friction between two or more rotating surfaces.

The torque transferred by the clutch is proportional to the clutch pressure, P_{clutch}, which is controlled by the control unit. The control signal is a value in the range [0,1], where 1 implies full pressure and thus maximum torque transferred.

The maximum pressure required in order to be able to transfer maximum torque from the engine is calculated by inverting the clutch model[1]. This means that the clutch is able to transfer this torque independent on the clutch parameters, which are set to the default values in Simulink.

This component also features a clutch slip sensor, making the difference in angular speed between the input and output shaft available to other parts of the model.

[1]See the help section in Simulink.

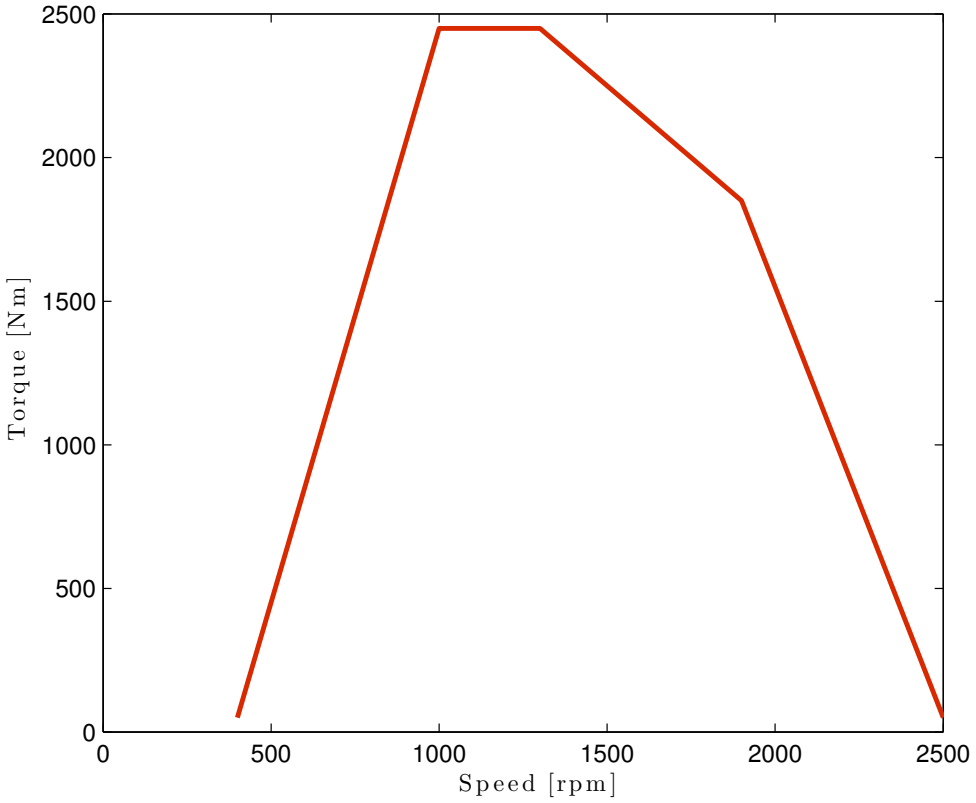


Figure 4.6: Available engine torque.

Gearbox

The gearbox is based on a simple gearbox and does not take any losses into account. The block is modified making the gear ratio an input and thus variable over time. This ratio is controlled by the control unit and given to the gearbox as an absolute value.

The final drive, often present in the rear axle in a truck, is also included in the gearbox model with ratio η_{final}. Thus, the actual gear ratio that is used in the gearbox becomes

$$\omega_{\text{in}} = \eta_j \eta_{\text{final}} \omega_{\text{out}}, \tag{4.6}$$

where η_j is the gear ratio for gear j, ω_{in} and ω_{out} is the input and output shaft angular speed respectively.

The gear ratios are partly obtained from Scania (gears 3-7) and partly tuned (gears 1 & 2) with the aim to get a realistic model. These ratios used are shown in Appendix D. η_{final} is set to 2.71 [**?**].

Brakes

A brake model is connected between the gearbox and the rest of the drivetrain. This means that the brake is modeled as acting on the main shaft instead of the wheel hubs which is the case in a real truck. The brake system is modeled using a disk friction clutch, where one of the physical ports is connected to the mechanical rotational reference and the other is connected to the shaft in the drivetrain network. Hence, the brake torque is

proportional to the disk pressure which is controlled by the velocity controller as a value in the range [0,1], 1 implying maximum pressure.

The maximum brake torque is determined from a user defined maximum retardation of the vehicle, $\dot{v}_{\text{ret,max}}$:

$$T_{\text{brake,max}} = m\dot{v}_{\text{ret,max}}r, \tag{4.7}$$

where m is the vehicle mass and r is the wheel radius.

This is then inserted in the inverted disk friction clutch model to get the required maximum pressure, similar to how it is done in the clutch model.

Default values in Simulink are used for the parameters in the clutch model, while $m = 12000$ [kg] is obtained from Scania and $\dot{v}_{\text{ret,max}} = 14$ [m/s^2] is tuned in order to get the truck to work well among the miniature trucks in the system. The wheel radius is set to $r = 0.495$ [m] [**?**].

Vehicle body

The vehicle body is a subsystem which takes into account the rolling resistance, air drag and inertia of the vehicle. The main shaft is connected to the input port of this block and a differential which transfers the torque to the rear wheels.

The wheels are modeled without slip and the rolling resistance on each wheel, F is considered to be directly proportional to the normal force, N so that

$$F_{\text{roll}} = N\mu, \tag{4.8}$$

where μ is the rolling resistance coefficient.

The air drag is defined as

$$F_{air} = \frac{1}{2}\rho A C_d v^2, \tag{4.9}$$

where A is the frontal area of the vehicle, ρ is the air density, C_d is the drag coefficient and v is the vehicle velocity.

These resistances are used together with the torque from the main shaft and the vehicle mass to determine the acceleration/deceleration of the vehicle.

In this part the numerical values used are $\mu = 0.007$, $A = 10.96$ [m^2] and $C_d = 0.6$ [**?**]. ρ is defined by Simulink as a default value.

4.2.2 Control unit

In order to model a semi automatic gearbox there must be a control unit that automatically controls the different components in the drivetrain. The main task of this part of the model is to take care of gear shifts and make sure that the clutch is applied properly when launching the vehicle from standstill. This subsystem also handles the applied throttle to the engine, e.g. during idling or when shifting.

This section will describe the different parts of the control unit and how it is implemented in Simulink.

Shift control

It is in this subsystem the gear shifts are initiated. It can be split into one logical system that initiates gear shifts and another that updates the gear state, G, which contains information of the desired gear and whether a gear shift is in progress. The latter system

also updates the shift direction state, (U/D), which explains whether shifting up or down. The possible values of these states are explained in Table 4.1 and Table 4.2.

To enable clutch handling during low speeds or start from standstill, a clutch control (CC) mode is available in the system. This function will be explained later on.

Table 4.1: The possible values on the state G and their meaning. N is the number of gears available.

G	Meaning
0.5	Clutch control mode
$\{1, 2, \ldots, N\}$	Gear G in place
$\{1.5, 2.5, \ldots, N - 0.5\}$	Shift in progress between $G + 0.5$ and $G - 0.5$

Table 4.2: The possible values on the state U/D and their meaning.

U/D	Meaning
1	Shifting up
-1	Shifting down

Depending on the current value of the gear state the first system uses engine speed, measured clutch slip and information from the gearbox control system to determine whether the gear state needs to be changed. The logic is explained in Figure 4.7, where n_{up} and $\hat{n}_{\text{down}}$ are the thresholds for up- and downshift respectively. The lower threshold is defined as

$$\hat{n}_{\text{down}} = \begin{cases} n_{\text{clutchout}} & \text{if } G = 1 \\ n_{\text{down}} & \text{otherwise} \end{cases}. \tag{4.10}$$

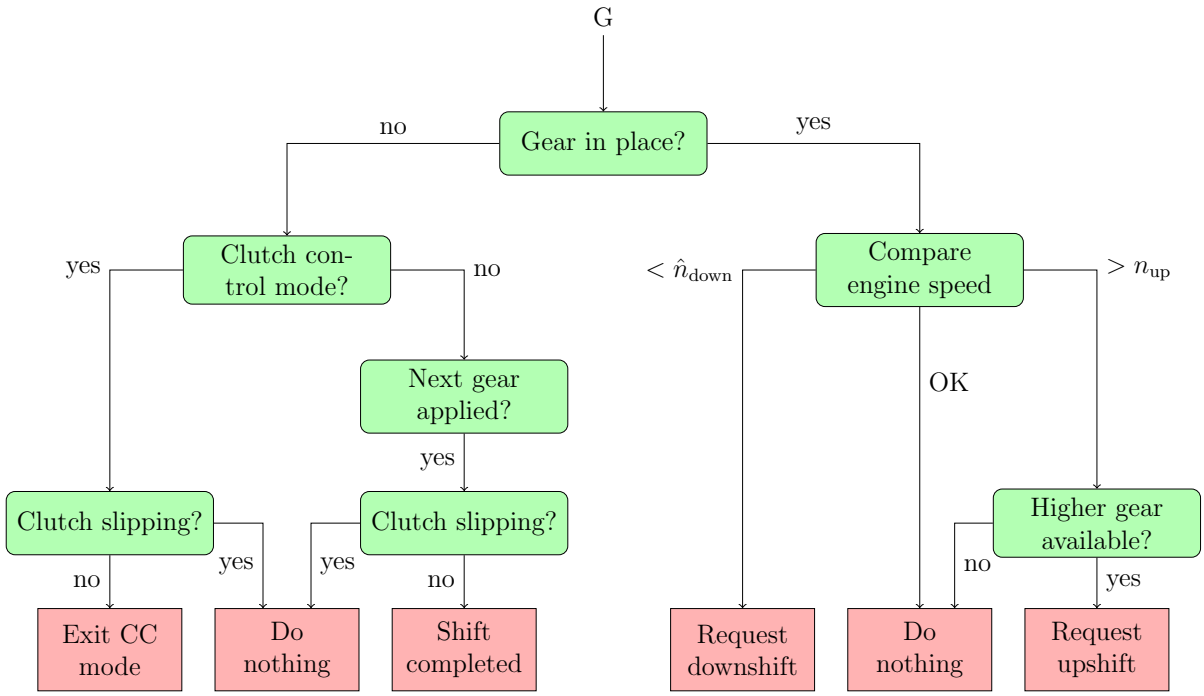


Figure 4.7: Description of shift decision logic. This algorithm is looped within the system when driving.

In the second part of the shift control system the gear state is updated. Signals from the above described logic, such as shift requests, are read and used to update the gear state. The workings are described in Table 4.3.

Table 4.3: Description of how the gear state is updated for different logic signals.

Signal	New G value	New U/D value
Upshift request	$G = G + 0.5$	$(U/D) = 1$
Downshift request	$G = G - 0.5$	$(U/D) = -1$
Exit CC mode	$G = G + 0.5$	$(U/D) = 1$
Shift completed	$G = G + 0.5(U/D)$	No update

The above described logic handles one shift at a time when driving normally. However, when braking hard ($-\dot{v} > \dot{v}_{\text{ret,th}}$) the shift control system is not able to handle the deceleration, since it has to wait for one downshift to initiate the next. To solve this, an override function is implemented so that the gear state is overridden and clutch control mode is entered when braking hard or when the engine speed becomes too low ($n_{\text{engine}} < n_{\text{idle}}$). Once this override function is enabled, the desired gear is determined from the vehicle velocity. This gear is then applied when going back to normal driving mode.

Values on $n_{\text{down}} = 850$ [rpm] and $n_{\text{up}} = 1500$ [rpm] are identified from data, obtained from Scania, logged while driving an actual Scania truck. $n_{\text{idle}} = 500$ [rpm] is obtained from Scania. $\dot{v}_{\text{ret,th}} = -2.8$ [m/s^2] and $n_{\text{clutchout}} = 700$ [rpm] are found by tuning in the model.

Clutch & Gearbox management

This subsystem determines the actual control signals that are sent to the drivetrain, i.e. the normalized clutch pressure and the actual gear ratio. Its functioning can be divided into three cases: the first when in clutch control mode, the second when shifting and the last when a gear is in place:

Clutch control mode: The clutch pressure depends on the demanded throttle and the engine speed. Once this mode is entered the normalized clutch pressure is increased by a first order system in which the time constant depends on the demanded throttle, δ_{dem}. However, δ_{dem} is overridden to zero if the engine speed falls below a user defined threshold, $n_{clutchin}$.

Also, if the engine speed falls below another threshold, $n_{clutchout}$ the normalized clutch pressure is set to zero.

$$\bar{P}_{\text{clutch}}(s) = \begin{cases} \dfrac{10\hat{\delta}_{\text{dem}}^2}{s + 10\hat{\delta}_{\text{dem}}^2} & \text{if } n_{\text{engine}} > n_{\text{clutchout}} \\ 0 & \text{otherwise} \end{cases}, \tag{4.11}$$

where

$$\hat{\delta}_{\text{dem}} = \begin{cases} \delta_{\text{dem}} & \text{if } n_{\text{engine}} > n_{\text{clutchin}} \\ 0 & \text{otherwise} \end{cases}. \tag{4.12}$$

Shift in progress: The clutch pressure is initially set to zero. The next gear is then applied and the normalized clutch pressure is ramped up linearly towards one. The time it takes to shift is T_{shift} [s]. The timings can be seen in Table 4.4, where $\hat{t} = \frac{t_{elapsed}}{T_{shift}}$.

Table 4.4: The clutch and gear control procedure during a gear shift.

$\hat{t}$ [s]	Action
$\hat{t} = 0$	$\bar{P}_{\text{clutch}}$ is set to zero
$\hat{t} = 0.25$	Next gear applied
$0.5 < \hat{t} \leq 1$	$\bar{P}_{\text{clutch}}$ is linearly increased from 0 to 1

Note that the shift time may be shorter than T_{shift} since the shift control system will consider the shift to be complete as soon as the gear is applied and the clutch slip is zero.

Gear in place: The normalized clutch pressure is always set to be one so that the gearbox is fully coupled to the engine allowing maximal torque transfer.

Numerical values are $n_{\text{clutchin}} = 1000$ [rpm], which is tuned in order to get the model to work well, and $T_{\text{shift}} = 1$ [s], which is identified in the data received from Scania.

Engine management

The last part of the control unit is the engine management, in which the actual applied throttle signal, $\hat{\delta}$, is determined. The desired throttle level, δ, from the cruise controller is overridden if braking or shifting. It is also overridden when the idle speed controller is

active, i.e. when the engine speed is low enough and the desired throttle signal is too low to maintain engine speed.

The idle speed controller determines the required throttle level to maintain the engine speed from the available engine torque, $T_{max}(n_{engine})$, and the engine friction, $T_{friction}$ so that

$$\delta_{\text{idle}} = \min\left[\frac{T_{\text{friction}}}{T_{\max}(n_{\text{engine}})}, 1\right]. \tag{4.13}$$

This is then compared to the desired throttle level to determine the applied throttle signal:

$$\hat{\delta} = \begin{cases} \max\left[\delta_{\text{idle}}, \delta\right] & \text{if } n_{\text{engine}} \leq n_{\text{idle}} \\ \delta_{\text{dem}} & \text{otherwise} \end{cases}, \tag{4.14}$$

where

$$\delta_{\text{dem}} = \begin{cases} 0 & \text{if (braking OR shifting OR idling)} \\ \delta & \text{otherwise} \end{cases}. \tag{4.15}$$

4.2.3 Velocity controller

The velocity controller determines the desired throttle level δ and brake level β. This is done using a PI controller with a reference speed, v_{ref}, received from the high level control in the system.

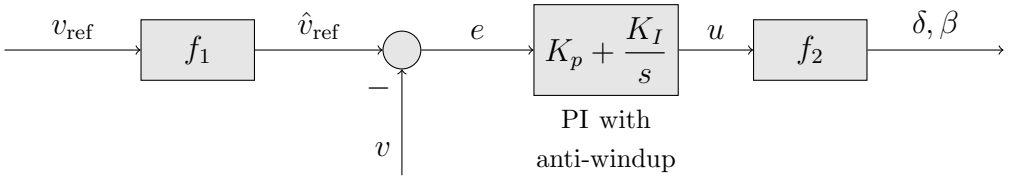


Figure 4.8: The velocity controller. v is the actual velocity, e is the error, K_p and K_I are the tuning parameters in the PI controller and u is the control signal.

Limitations in the model prevent the use of too low velocities due to difficulties in handling the clutch slip that is required if the velocity is too low to drive on first gear with the clutch fully coupled. This, together with a upper limit on the velocity, is handled in the block symboilized by f_1 in Figure 4.8 according to

$$\hat{v}_{ref} = \begin{cases} 0 & \text{if } v_{ref} < v_{\min} \\ \min[v_{\text{ref}}, v_{\max}] & \text{otherwise} \end{cases}. \tag{4.16}$$

The output signal, u, is saturated and split up into the throttle and brake signal in the block denoted f_2 in Figure 4.8 according to

$$\hat{\delta} = \max[0, u] \tag{4.17}$$

$$\beta = \min[0, u] \tag{4.18}$$

$$u \in [-1, 1]. \tag{4.19}$$

The PI controller also includes an anti-windup functionality using back-calculation.

The values used are $v_{\max} = 110/3.6$ [m/s] and $v_{\min} = 2$ [m/s]. The controller parameters $K_p = 0.8$ and $K_I = 0.2$ together with the back-calculation coefficient $K_b = 1$ are found by manual tuning of the controller.

4.2.4 Steering kinematics

The steering kinematics in the real truck model are the same as for the miniature trucks, see subsection 4.1.3, with parameters defined individually for this model. Numerical values can be seen in Appendix D and are tuned with the aim of a realistic model in mind. An exception is the length between the front and rear axle L, which is measured on the miniature trucks in SML and scaled to a full size truck.

4.2.5 Limitations

The aim of this model is to capture the dynamics of the overall vehicle, with realistic behavior regarding acceleration, braking and gear shifting. With this in mind, some simplifications are made.

- The torque from the brakes is assumed to act on the main shaft instead of at the wheels as in real trucks.
- Trucks are usually fitted with additional brake systems, such as retarder and exhaust brake. These systems are not included in this model.
- The gear shift logic is based on the engine speed alone. It is often more complicated in a real truck where the decisions are made taking more factors into account, such as road gradient and train weight.
- There is a limitation on the velocity that prevents the truck from driving at velocities that are lower than the lowest possible velocity on the first gear with the clutch fully coupled.
- The maximum retardation when braking, $\dot{v}_{\text{ret,max}}$, is too high compared to the real case. The higher value had to be chosen to make the model work well in cooperation with the miniature trucks, which can slow down very fast on the simulated road.

Chapter 5

Low Level Control

Low level control deals with the vehicles own controllers such as trajectory following and velocity.

5.1 Velocity control

For all trucks in the scenarios there exists a velocity controller. As explained in the section above, the Scania truck model on the xPC has a PI plus saturation included into the model. For the miniature trucks and their simulations the controller is located in LabVIEW embedded in "Controller_FIR.vi". Basically, it has the same PI plus saturation setup as for the Scania model, but without anti-windup and different parameters, which can even be accessed via the control panel during runtime.

5.2 Trajectory Following

In this section the algorithm to calculate a steering angle from the given road and vehicle waypoint is described. The vehicles are supposed to follow a given road, which is represented by a set of points in LabVIEW. In previous steps in the program, the vehicle position, from the motion capture system or the simulation output, is related to the nearest waypoint ahead. If waypoint location and vehicle position do not match exactly, the angle from the vehicle orientation to the assigned waypoint can be used to calculate a proportional steering command. This simple approach already gives a working trajectory following. Since the steering is always calculated based on the next point, the steering signal can change rather rapidly, which results in a hard steering behaviour. Therefore, another approach was implemented, which bases the steering decision not only on the angle to the next waypoint, but is looking a number of waypoints ahead. The steering command then becomes a weighted sum of the future angles. As a result, the vehicle starts to steer earlier when it reaches a turn and less aggressive. These result has not been verified by measurements, but is a general observation. Detailed evaluation of different weighting functions could be a (sub)task for the future. An illustration of the algorithm is shown in Figure 5.1.

In the lower left corner the vehicle is seen with its initial orientation. The first angle considered is the angle between the vehicle yaw and the line 'vehicle position to assigned waypoint'. Afterwards, the angles along the trajectory are calculated. The steering

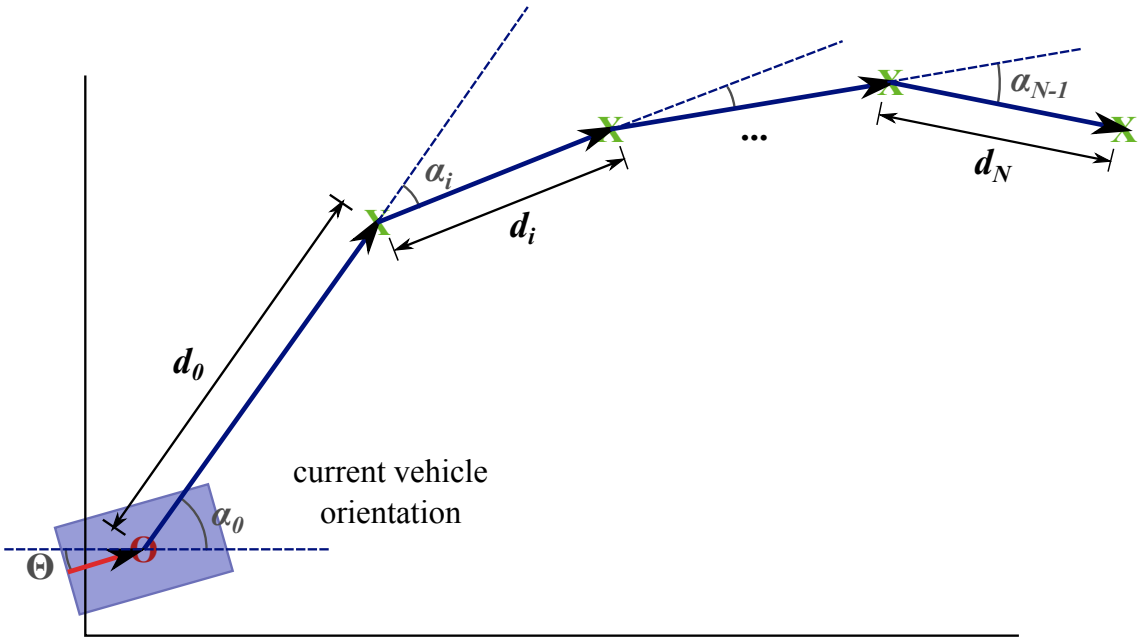


Figure 5.1: Illustration of the controller for road following. The steering command is calculated by a weighted sum of the future trajectory angles.

command is determined by

$$\phi = c \cdot \sum_{i=0}^{N-1} w_i \alpha_i, \tag{5.1}$$

where the factor c and the weights w_i, as well as N, the number of waypoints to look ahead, remain degrees of freedom. In the version which was handed over, the weights are a combination of distance based weights and index based weights. The latter ones are given by

$$w_{i,1} = \frac{N-i}{\sum_i^{N-1} i} \tag{5.2}$$

and ensure that the influence of angles decreases when they are further away. The distance based weights are

$$w_{i,2} = \frac{d_i}{\sum_i^{N-1} d_i} \tag{5.3}$$

and increase the influence of angles that are important for a long part of the trajectory. This is especially important if the vehicle loses position, or starts off-track. In these situations the vehicle is far away from the trajectory and just the first angle will have a relevant weight. The vehicle then attempts to drive straight back to the assigned waypoint.
To combine two weighting functions, they are multiplied and normalized again afterwards. Without the normalization, the steering response will usually be out of the intended range, i.e., depending on the underlying function, almost no steering or extremely aggressive steering.
The implementation of the steering controller is done in a mathscript inside the LabVIEW program. The corresponding SubVI is "FIR_steering_controller.vi". For the calculation

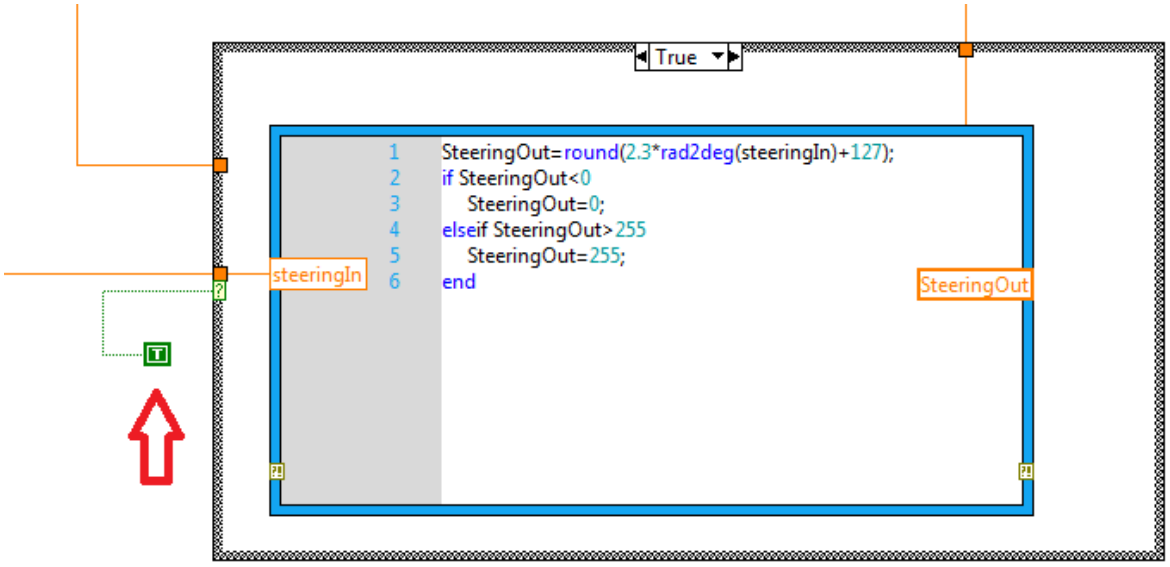


Figure 5.2: Activation (T), or Deactivation (F) of the FIR steering controller. On deactivation, the old controller is used instead.

of the first angle an additional position on the vehicle is calculated to use the cross-product and avoid trouble with different angle representations and non-linearities, e.g. $\phi \in [0, 2\pi)$ or $\phi \in [-\pi, \pi)$ and $\phi = \phi \bmod 2\pi$. The controller can be (de)activated in "Controller_FIR.vi" by setting the boolean flag indicated in Figure 5.2.

The idea of the steering controller is inspired by FIR, *finite-impulse-response*, filtering. In this application the weighting function is basically estimating the curvature ahead. Other approaches could be to use an interpolation that looks forth-and-back, i.e. basing the decision on a few waypoints behind the vehicle and some in front. Additionally, the many degrees of freedom are assumed to solve more complicated steering situations, e.g. if a trailer is added and has to follow through a curve. Weighting the last few angles negative would result in a lower steering when the curve is entered and could help keeping the trailer on the lane.

Chapter 6

High Level Control

High level control, or traffic control, enables the vehicles to cooperate and/or avoid collisions. In our system the only control is the velocity of the vehicle, i.e. sending a new reference velocity to the low level controllers of each vehicle. This is due to the scenarios only could include one lane roads. Future work could include roads with several files and hence control of the steering as well.

In this years project three levels of complexity are developed and implemented. The first and simplest one, is a *centralized control* with the implementation of traffic lights to control the traffic flow. The second scenario is called *decentralized control* and is based on pre-defined traffic rules, example given, the right-hand rule. The most complex scenario is the third one, denoted *cooperative decision*, where vehicles agrees on what action they should take in order to avoid collisions and this complexity is explained in the end of this section together with the concept of vehicle-to-vehicle communication.

6.1 Centralized and Decentralized Traffic Control

There are many tools to use in order to control traffic and interaction between vehicles. For example traffic lights, signs and rules is used. In this section the different signs and lights will be described and as mentioned earlier, the traffic lights are used in the *centralized traffic control* scenario. *Decentralized control* using the right-hand rule is also explained.

6.1.1 Traffic Rules Hierarchy

The different traffic rules have different priorities in the following decreasing order:

1. Traffic lights
2. Traffic signs
3. Traffic rules

6.1.2 Traffic Lights and Traffic Signs

As Table 3.1 shows, the information about the traffic lights and traffic signs is read from the data file *SignInfo.txt*, created by the *Road_Network_Creator* tool. The traffic lights are placed at a given waypoint in the traffic direction (as in real-life) and as shown in

Figure 6.1. It is those waypoint numbers that are used in order to state to the vehicle where they should start reacting to the given light or sign.

The implementation is easily done and tested on a crossing, since it is there where most traffic control is needed, e.g. for avoiding collisions and traffic congestion.

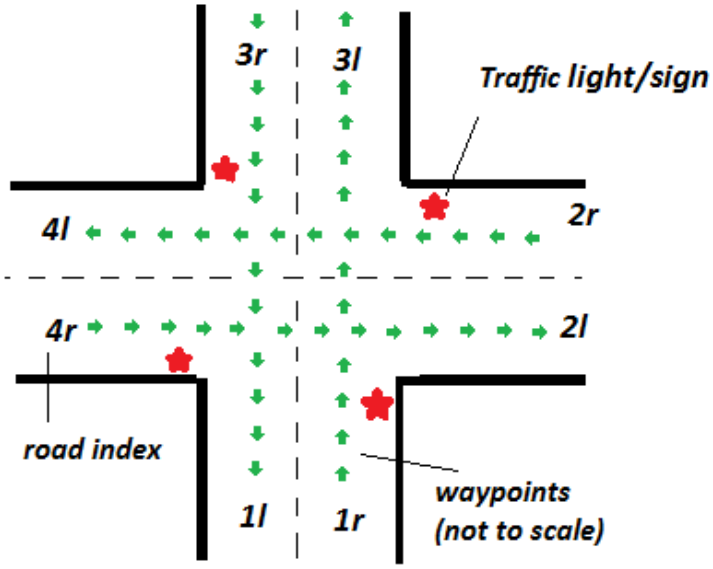


Figure 6.1: Sorted crossing and placement of traffic lights/signs

6.1.3 Traffic Lights

The traffic lights have the highest priority of all traffic rules. When a truck is approaching a red traffic light and it gets within a certain distance it sets its reference velocity v_{ref} to 0, thus stopping.

The vehicle waits for the light to turn green before continuing. When the light is green, lower-priority traffic rules set in, as the vehicle will be allowed to drive as long as no traffic rules in that segment is violated. This is the case, for example, when there is a speed limit sign before or just after the green light is passed.

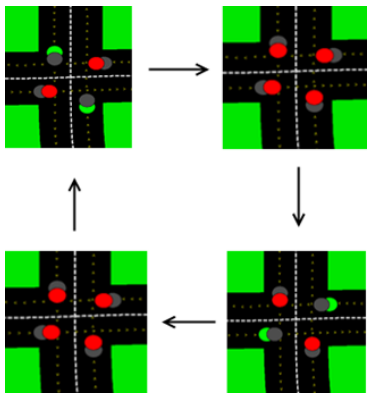


Figure 6.2: Alternating of traffic lights in a crossing

In the crossing scenario, traffic lights are in use. This scenario includes four automated lights that are alternating. They are coupled two-way, so two lanes that do not cross have green at the same time. The traffic lights can only have the colour green or red. To add a safety margin, as in real traffic, all the lights will be red for a short period before alternating to green.

6.1.4 Traffic Signs

Traffic signs implemented in this project are speed limit, yield and stop.

Speed Limit Sign

The speed limit sign sets a maximum velocity a truck can have. This is done by the truck looking back on the same road and sets its maximum speed to the last speed sign. The speed limit sign is used in all scenarios.

Yield Sign

The yield sign, is used so that a vehicle coming from a minor road onto a highway, for example, would have to check if there is any vehicle coming from the left or right before engaging on that highway.

The yield sign is also implemented as a part of the highway scenario to mark the beginning of a ramp.

Stop Sign

The stop sign is mainly implemented in crossings. It works so that when the vehicle reaches the stop sign, the reference velocity v_{ref} is set to 0.

It is thus similar to the red traffic sign, except that for the stop sign, after the full stop, the yield sign sets in, and the vehicle must check for left- and right coming traffic before it may continue.

6.1.5 Right-hand Rule

The right-hand rule is implemented in crossings, so that the vehicle approaching from the right always have the priority to drive. As stated previously with the creation of the *crossingMatrix* (see also Figure 6.1) the crossing has been sorted so vehicle always knows what is on its right.

The *crossingMatrix* is sorted such that the coordinates of the road on the right are placed in the columns on the right. In other words, it is similar to the road-indexing in Figure 6.2, 4 - 1 - 2 - 3 - 4 - ... For example, when coming from 3r, the road on your right will be 4, and so on.

The first iteration in the implementation of the right-hand rule is to check the vehicle's own position with respect to the distance to the crossing.

Second, the position of every vehicle in the network is checked, since it is supposed to be so that information on each vehicle's position is accessible by all other vehicles. If any of them is on the road on the vehicle's right, the distance from the crossing is calculated. Here for the case of the yield sign section 6.1.4, the left side is also check on the same manner.

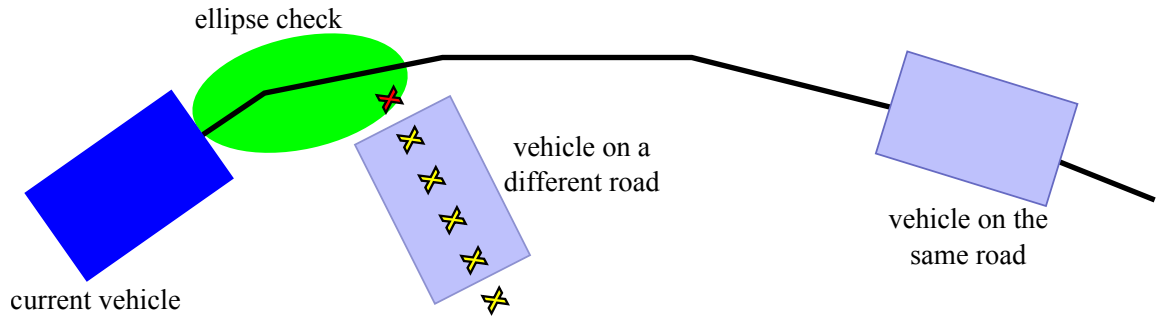


Figure 6.3: Black line: trajectory up to look-ahead distance, blue: vehicles, green: elliptical stop zone in front of the current vehicle, 'x': points representing a vehicle.

Third, a decision is made: if the vehicle can manage to pass through the crossing before the other vehicle on your right does, then it keeps driving. Otherwise, if there is a risk that the two vehicles meet in the crossing, the priority to drive is given to the vehicle on the right. The own vehicle's speed is set zero until the crossing is fully cleared. For the case of a yield sign, the sight has also to be cleared for any left-coming traffic.

6.2 Collision Avoidance

Safety is maybe the biggest requirement for autonomous vehicles. Whether considering the optimization of the traffic flow or the safety of people, a collision comes at high cost. The collision control is a task that is fundamental and has to work reliably. With collision control, prevention by adapting speed or emergency braking is meant. Obstacle avoidance by steering is often associated with collision control as well, but will be omitted, since it causes difficult problems along with the traffic representation in the simulation.

The collision controller is called for each vehicle and checked against each other vehicle with valid positions in two steps. Figure 6.3 indicates the elements of the explanation. First, if another vehicle's waypoint number corresponds to a waypoint number ahead of the current vehicle and they are on the same lane, the speed is lowered if necessary - if the vehicle in front is faster, the current speed is kept. The adaptation of speed allows vehicles to form queues and line up on the road with some spacing. However, this method does not see vehicles coming from the side, e.g. in a crossing, since the waypoint numbers are far apart or the lanes are different. Therefore, a second step may override the first one. In the second step, a number of points on all vehicles are calculated from their respective position and the yaw angle. In front of the current vehicle an ellipse is placed, which has roughly the size of a vehicle and is rotated according to the road curvature. It is then checked, if one of the positions lies inside the ellipse. If so, the current vehicles reference velocity is overwritten to zero.

An ellipse is maybe not the most intuitive choice, but it has several beneficial properties. It fits smoothly into curves, has a mathematical representation which is easy to handle and with center point and two radii only few parameters.

In the implementation the ellipse is used in the matrix representation

$$\mathbf{x}^{\mathbf{T}} \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} \mathbf{x} = 1, \tag{6.1}$$

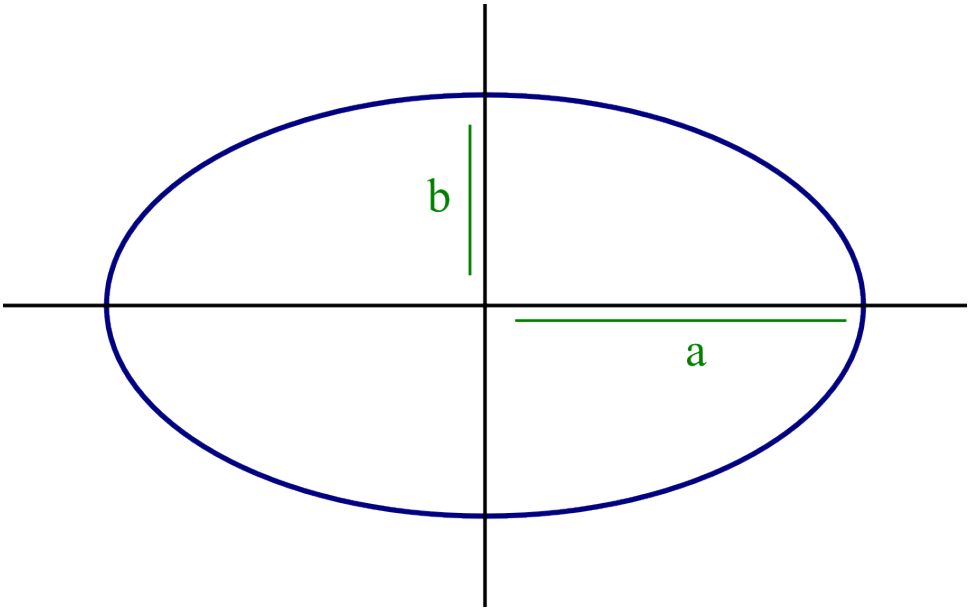


Figure 6.4: A standard ellipse, with the radii a and b.

where a and b are the measures for the ellipse according to Figure 6.4. Note that the equality to one denotes the ellipse itself, a less-than the inner and a bigger-than the outer region of the ellipse. With the matrix representation, operations like rotation and translation are easy to apply. If the equation is evaluated to be less-than one for a translated and rotated point, the point is inside the ellipse and the system has to react. Let p be a point to test, t the translation of the ellipse and $R(\phi)$ the rotation matrix

$$R(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}, \tag{6.2}$$

then the equation to test is

$$(\mathbf{p} - t)^T R^T \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} R\,(\mathbf{p} - t) \leq 1. \tag{6.3}$$

To save some computation the rotated ellipse can be calculated once as

$$F = R^T \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} R \tag{6.4}$$

and reused for all checks with the respective truck. In the implemented version, the ellipse is placed based on the waypoint assigned to the vehicle and takes the angle to the next waypoint into account. As mentioned, the collision controller is an emergency system which may overwrite reference velocities from the high level control. As such it is the last system in the control trail and is called right before the velocities are sent to the vehicles.

Although a trailer model was developed, the related changes in the trajectory following and collision control algorithms were considered too time consuming or this project. Other concepts of collision control can be based on different shapes or vehicle representations. A known and unidentified problem occurs if a vehicle comes from the side and has another vehicle blocking its path. If the second vehicle gets very close to the driver cabin, e.g. due to inertia, the collision zone is 'underrun' and the vehicle starts driving again. Since this case rarely appears, the solution was still considered sufficient for the lab.

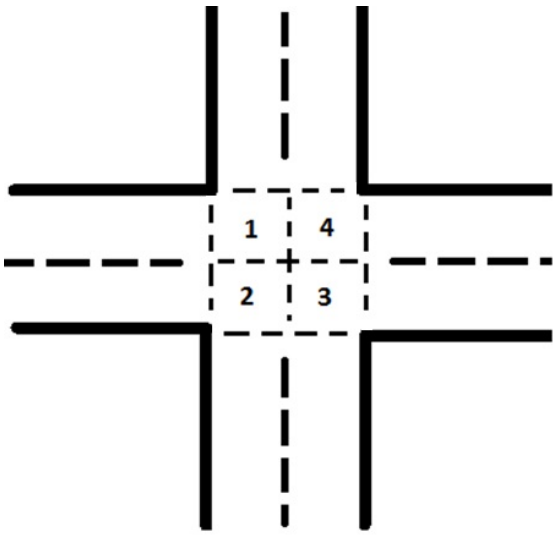


Figure 6.5: Sections in crossing.

6.3 Vehicle-To-Vehicle Communication

Each vehicle can broadcast information about its velocity, heading and intended action. Other vehicles can then collect this information and return their own data. Together with information about their states they can also send other type of information so they will be able to cooperate and solve traffic situations. This is called vehicle-to-vehicle communication (V2V). There has been two major scenarios implemented as a part of the traffic control. The first scenario deals with crossings, while the second scenario deals with highway entrances.

How V2V is implemented in the different scenarios is described below.

6.4 Crossing Scenario

The first scenario, the crossing scenario, is about trucks driving through an intersection depicted in Figure 6.5. The trucks will have to communicate with each other in order to adapt velocities so both collisions and complete stops is avoided, to make the traffic smooth.

To achieve this, the crossing has been divided into four sections numbered from 1 to 4 in an counter clockwise direction. Dividing the crossing into sections makes it possible to detect which parts of a crossing trucks will use.

6.4.1 Information exchange

When a truck is approaching a crossing and gets within a certain distance it will start sharing information with other trucks that are also approaching the same crossing. The truck will then continuously broadcast the data shown in Table 6.1 until it is past the crossing.

Row 1 to 4 in the shared collision matrix can take the value 0 or 1, and specify which parts of the crossing will be used. Each truck will determine which sections it will use based on two inputs, the entry section and the designated direction. An example of which sections will be used when the entry section is four can be seen in Table 6.2.

Table 6.1: Shared collision matrix.

Information	Type of data
Section 1	Boolean
Section 2	Boolean
Section 3	Boolean
Section 4	Boolean
Time to crossing	Seconds
Time until I am out of the crossing	Seconds
Entry section	Integer, 1-4

Table 6.2: Example of used sections, entry section = 4

Truck heading	Occupied sections
Right turn	4
Forward	4 and 1
Left turn	All sections

If the distance between two trucks are in a certain interval they are considered to be driving in a convoy. The last piece of information a truck will broadcast is whether it is a part of a convoy or not. This is later used for prioritizing the driving. The longer a convoy is, the higher the priority it will have. Each truck will therefore broadcast the total number of trucks in its own convoy and the ID of the last truck.

6.4.2 Algorithm

The algorithm works in such a way that each truck looks for potential collisions. This is based on the information exchange seen in Table 6.1. If a potential collision is detected, one of the involved trucks will slow down as no truck is allowed to increase its velocity.

The collision detection works as follows: each truck will first check if it will use at least one of the same section as another truck. If this is true a convoy length comparison will occur. Depending on the convoy lengths, different actions will be taken, but it will always be the shortest convoy that decreases speed. The shortest convoy will adapt its velocity so that it will reach the crossing when the last truck in the longer convoy leaves the crossing, so a collision is prevented while the traffic flow is kept.

If two convoys of equal length will use the same section, the convoy with smallest time to the crossing will have priority. Trucks that are not close to other trucks are considered being in a convoy with itself. Theses convoys have length 1 and their own ID as last truck. This is needed for the algorithm to work in all cases.

When a collision is detected, a truck will have to slow down. The new velocity is adjusted so that the yielding truck will reach the crossing some time after the other truck has exited. This is based on the timing seen in Table 6.1. The new velocity is based on the old velocity and a factor *'k'* ($v_{new} = k \cdot v_{old}$). The factor *'k'* is given by Equation 6.5, where *timeIncrease* is the time that needs to be added to the yielding truck's *timeToCrossing*.

$$k = \frac{\text{myTimeToCrossing}}{\text{myTimeToCrossing} + \text{timeIncrease}} \tag{6.5}$$

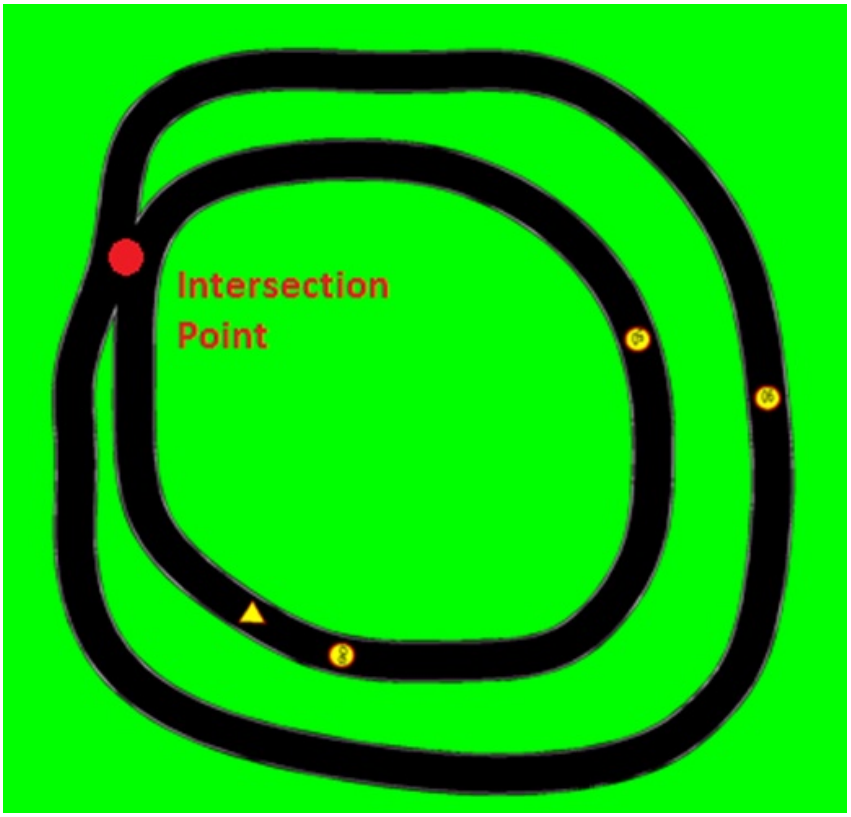


Figure 6.6: Highway entrance map. Outer road is highway road while the inner road is a regular road. The intersection point marks the entrance point.

6.5 Highway Entrance Scenario

The highway entrance, the second scenario, is about trucks entering a highway from a regular road. The highway road holds a higher velocity than the other road, and trucks on the highway are not allowed to stop, meaning that there is a minimum velocity for the highway trucks. The objective for this scenario is to have the trucks react to other trucks in order to achieve a good traffic flow when entering the highway.

The scenario is implemented as a separate Mathscript element within the main LabVIEW code, which acts as the decision maker if the condition for a valid highway entrance is found. The condition for an entrance is to have a yield sign followed by a single transition point to a different road within a set distance from the sign, as well as having the script activated from the main control panel in LabVIEW. The current implementation supports both V2V cooperation (between trucks on highway and trucks on regular road), and full yielding by the trucks on the regular road. However, the main focus for this project regarding this scenario is the vehicle-to-vehicle communication, meaning that this section will mainly cover that complexity.

Figure 6.6 shows the map, created with the *Road_Network_Creator*, which has been used for testing the concept of the highway entrance. The highway road is represented by a single-lane road going on the outside of the regular road. Currently, the implementation does not support the use of a secondary highway road lane for collision avoidance through lane-switching, which is why the highway road is represented by a single-lane road.

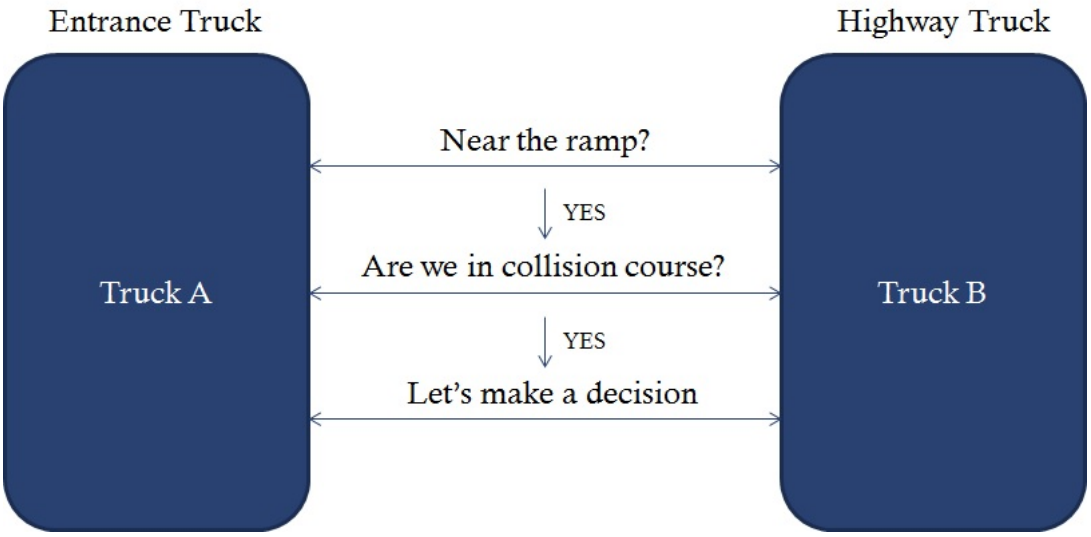


Figure 6.7: Flow chart showing the general communication concept.

6.5.1 Information exchanges

For vehicle-to-vehicle communication, the information given between trucks consists of their own timings until reaching the entrance point, as well as a flag in case they are breaking for another truck. Therefore, each truck must have certain information about itself, which can be attained through a GPS-receiver together with a GPS-map (represented by the motion-capture system installed in the Smart Mobility Lab) on each truck. A truck can then calculate its time until the entrance point based on distance to the entrance and its own velocity. The information can then be broadcast to other vehicles in the vicinity e.g. through Wi-Fi in order to determine their situation in regards to other vehicles.

The information exchange specific for the entrance scenario is restricted to be between trucks on the highway and the regular road.

Figure 6.7 shows a conceptual flow chart for the communication flow between trucks.

6.5.2 Algorithm

The algorithm can be described as a customized anti-collision system where the trucks look for potential collisions and then try to solve it. Potential collisions are detected through the timing information from each truck in the vicinity, which will change as their velocities are adjusted. A truck that receives a timing similar to its own timing will interpret it as a potential collision, this collision will then have to be resolved.

The first two steps in the communication are only to determine if any action has to be taken at all. If an action has to be taken, i.e. a potential collision has been found, then the two vehicles will have to make a decision upon which of the two trucks should slow down for the other truck. The truck that is designated to slow down will do so and immediately set a flag for it, so that this decision is not overwritten by other consecutively detected collisions. In case a truck detects multiple potential collisions it will attempt to solve them sequentially beginning with the collision that has the soonest estimated time until collision. The trucks will check this for every other detected truck in the vicinity.

To avoid a collision with trucks A and B, either truck A or B has to slow down, which

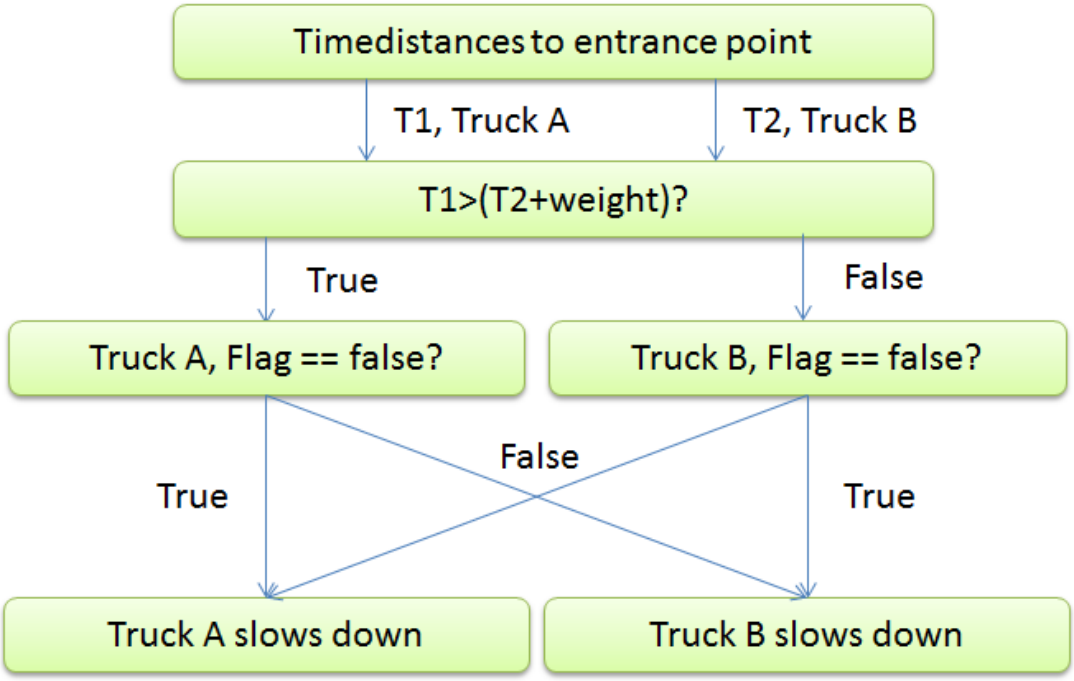


Figure 6.8: Flow chart showing the general algorithm concept.

is decided according to Figure 6.8. The truck that has to slow down is the truck with the greater timing until reaching the entrance point. However, this can be weighted to ones preferences (enables full yield for a road). To slow down a truck, its current velocity is scaled down by a factor *'k'* ($v_{new} = k \cdot v_{old}$) which is determined by the timing difference between the trucks, the desired time margin (time distance between two trucks), and the truck's own timing until reaching the entrance point according to Equation 6.6.

$$k = \frac{1}{1 + \frac{\text{DesiredTimeMargin} - \text{TimingDifference}}{\text{OwnTimingUntilEntrance}}} \tag{6.6}$$

With the adjusted speed the entrance truck will be able to enter the highway without risking a collision with another truck. Trucks on the highway are also influenced by the trucks' "convoying" mechanism from the regular anti-collision system (thus creating a queue line with certain spacings between the trucks), which will further help to smooth the traffic, since the gaps created good opportunities for the entrance trucks to enter the highway.

Most parameters relevant for this scenario can be adjusted from the main LabVIEW control panel.

6.5.3 Problems

Most problems related to the highway entrance scenario can be fixed by adjusting the highway entrance parameters in the LabVIEW main control panel. The parameters should be changed depending on how many trucks there are in the map, and how the map is designed.

A known issue is that the trucks on the entrance road might come to an almost halt when encountering multiple trucks on the highway near the entrance. This can occur if

there are too many trucks on the highway and the distance between the highway trucks is too small. The entrance truck might then come to a state where it will yield for all of the trucks until they have passed. However, this can also be avoided through changes in the main panel parameters, e.g. by increasing the desired timing distance between trucks. Although, increasing this parameter decreases the maximum number of trucks on the highway based on the length of the highway road, see Equation 6.7.

$$n = \frac{\text{RoadLength}}{\text{TimingDistance} \cdot \text{AverageRoadVelocity}} \tag{6.7}$$

Chapter 7

Conclusions

The main objectives of the project have been fulfilled and the results have been demonstrated. A HIL simulation environment including physical miniature trucks and virtual truck models with desired characteristics has been successfully established. More importantly, the platform is scalable in terms of changing traffic scenarios.

Within the simulation environment, traffic control algorithms have been developed and tested. With V2V communication, collision avoidance in a four way crossing and on a highway entrance ramp has been successfully demonstrated. In the crossing situation, however, it would have been more desirable to not have vehicles come to a complete stop and rather adjust each vehicle's speed so that no single truck comes to a stop. Due to limited time put on developing the algorithm, such was not achieved in the end. With a small number of trucks run in the environment though, this issue did not occur. Other types of traffic control, not using V2V, have been successfully implemented as well.

Due to workload and time limitations, a few demands from the project purchase order had to be reformulated. The changes were thoroughly discussed and anchored with the project supervisor and should not bee seen as a failure.

Consequently, the project was delivered on time and with sufficient quality. Time was probably the single most limiting factor and several improvements can be done in the future. The established HIL is a flexible platform on which others can improve and develop traffic control systems and vehicle communication algorithms.

Chapter 8

Suggestions for future work

8.1 Models and Control

8.1.1 Models

For future improvements in regard to the models, the following points can be considered:

- The gear shift logic in the real truck model could be improved to take more factors in to account when initiating gear shifts.
- In the real truck model, clutch handling on low speeds could be improved so that the truck is able to drive at speeds closer to zero.

8.1.2 Control

For future improvements in regards to control, the following points can be considered:

- In the collision controller, the first look-ahead step could be improved to not only react on other vehicles, but actively control the distance. This improvement could then be extended to solve real platooning tasks.
- The FIR steering controller has the weights and some overall factors as degrees of freedom. These should be used to experiment with more challenging scenarios, e.g. if trailers have to be steered through sharp turns. As well, other approaches than the look-ahead FIR could be tested, e.g. an interpolation between a few past and future angles.
- Due to the look-ahead distances in the collision avoidance and other stop scenarios, such as red lights and stop signs, the stopping distance of the real truck is too long. The different functions for stopping vehicles in different scenarios were based on the stopping distances of the miniature trucks, which is much shorter and unrealistic. This caused the real truck to run red lights and collide with other trucks.

  As a workaround, the maximum retardation in the real truck model was increased to a unrealistic value. This means that the real truck is able to brake harder than what is possible in reality, considering the scale. Suggestively, the function for stopping vehicles should be suited for the real truck to be part of the traffic with realistic braking dynamics. As an idea, this could be done by using custom look-ahead distances for the real truck.

8.2 Traffic Control

8.2.1 Crossings

For future improvements in regards to the crossing scenario, the following points were considered:

- If a truck for some reason stops while inside a crossing, then every truck which will use the same section will also stop, because a potential collision is detected. Because the truck will try to adjust the speed so that it enters the crossing when the stopped truck leaves after infinite time.
- If a truck is part of a convoy, it will broadcast the ID of the last truck. If a collision is detected comparison will always be between one truck and the last truck of a convoy. In the case of the last truck being too far from the crossing the timing information will be set as zero and therefore the comparison will be incorrect.

8.2.2 Highway Entrance

For future improvements in regards to the highway scenario, the following points were considered:

- Implement support for using a secondary lane on the highway for collision avoidance through lane-switching.
- Improve the entrance detection system. Suggestively by adding a feature to the Map Creation Tool that enables the possibility to directly mark an area for being a highway entrance.
- Improve the implementation of "vision-range", i.e. how far away each truck can detect other trucks. At the moment it is implemented as an area of effect with the entrance point as origin.

8.3 System integration

8.3.1 Visualization

The visualization can be improved in many ways. The feature of enabling and disabling the traffic signs in LabVIEW should make them appear respectively disappear from the visualization. A slot in the packet structure is dedicated for this but not implemented.

One should consider redoing the visualization tool in a more appropriate programming language such as C++. MATLAB has many disadvantages when building applications for generating animated graphics.

8.3.2 xPC Target

The two xPC Target machines/models have been introduced and included in the HIL platform. One represents the kinematics of the truck and the dynamics of vehicle components such as engine, gearbox and so on. The other xPC Target machine is then supposedly a

self-driving controller which gathers information about all neighboring trucks and road infrastructure, and maneuvers the truck autonomously.

The self-driving controller xPC Target model is, however, currently only passing reference values from the main LabVIEW program to the truck xPC Target machine. This is due to limited time in the project and the control algorithms implemented in LabVIEW could not have been translated to Simulink environment. Whomever takes on this project can then start implementing self-driving algorithms onto this other xPC Target machine, while leaving the truck model xPC Target as is.

Bibliography

[1] J. P. Alvito. Implementation of traffic control with heavy duty vehicle anti-platooning. Master's thesis, KTH, Automatic Control, 2013.

[2] M. Ammozadeh. *Trucks, Quadrocopters, and Mocap User's Manual*. KTH Automatic Control Lab, 2012.

[3] C. Elm. *System Architecture in a Heavy Duty Vehicle Platooning System using xPC Target*. KTH Automatic Control Lab, 2013.

[4] A. Hauksson, F. Svensson, B. Mengana, C. Westermark, J. Lycke, J. Sundberg, K. Imhäuser, and S. van de Hoef. Automatic control project course. Technical report, KTH - Royal Institute of Technology, December 2012.

[5] P. Lima and J. Alvito. *Mocap, Trucks and Visualization Tool User's Manual*. KTH Automatic Control Lab, 2013.

[6] P. F. Lima. Implementation and analysis of platoon catch-up scenarios for heavy duty vehicles. Master's thesis, KTH, Automatic Control, 2013.

[7] J. Nylander and E. Wäng. *User's Manual for Road Map Creation and Road Map Projection*. KTH Automatic Control Lab, 2013.

Appendix A

Purchase Order

The following is the purchase order given to the project group in the beginning of the course.

Purchase order #EL2421
Automatic control, project course, 2013

Jonas Mårtensson

We hereby order a hardware-in-the-loop simulation platform for cooperative vehicles according to the following specification.

Background

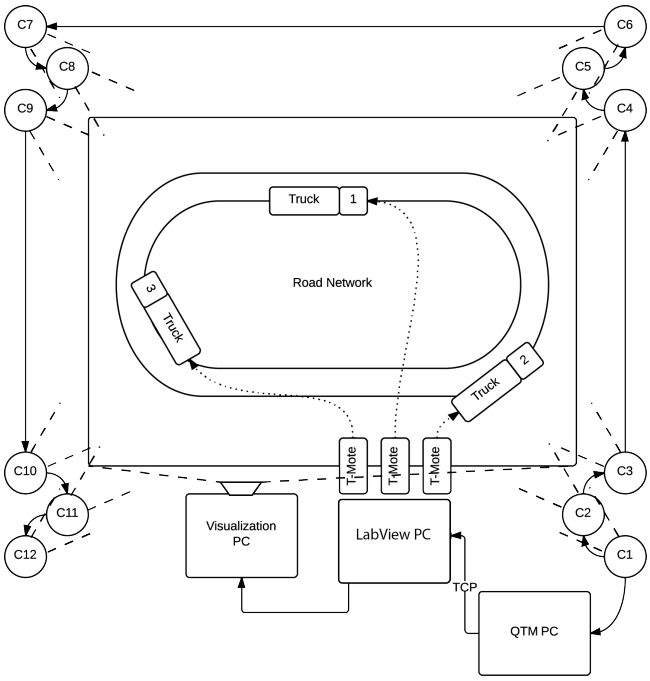
Today's sensor and communication technologies have enabled not only autonomous vehicles but also vehicles systems that communicate and cooperate. By communicating vehicle states (positions, velocities etc.), and deploying some protocol for interactions or negotiations, the vehicles can autonomously adapt to the surrounding traffic and the traffic system can be made safer, greener and more efficient. Such systems are studied in the Smart Mobility Lab at KTH, both with miniature vehicles in a lab environment and with full scale trucks in collaboration with Scania.

Miniature vehicle platform

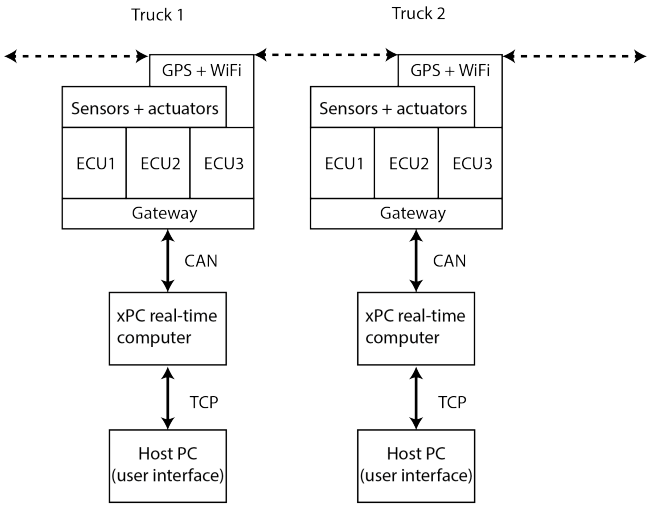
The miniature vehicle platform is based on scale 1:14 model trucks that are controlled over a wireless link from LabView running on a regular PC. Positioning is given from an infra-red camera system in the ceiling. The object tracking is performed in a separate computer (QTM PC) and the position and orientation of the vehicles are made available over the LAN network. The road network is generated from a third PC that also displays an image of the network on the floor using a projector in the ceiling. The road network is represented as series of waypoints with connection nodes at the intersections. Active traffic lights and signs can also be added to the road network.

Real vehicle platform

The Scania trucks are equipped with an electronic steering unit that actuates the steering angle and a cruise control unit that actuates the throttle and the brakes. Our interface to the control units are curvature and velocity references, which are sent over a gateway CAN bus (Controller Area Network). The trucks are equipped with GPS receivers, radar and WiFi to communicate with other vehicles and the road infrastructure. These data, along with onboard sensor data (e.g. speed), are also available from the CAN gateway. The control algorithms are implemented in Simulink and executed on a real-time computer running xPC target.



Figur 1: Miniature vehicle platform



Figur 2: Real vehicle platform

Task

Your task is to create an integrated hardware-in-the-loop simulation environment for the two platforms described above. Models of the fulls scale Scania trucks should run in real-time on a dedicated xPC target PC and models of the miniature trucks should run in real-time in LabView or Matlab. Both simulation models will be visualised on the floor with a ceiling-mounted projector that projects moving images on the floor. A scaling between the full scale and miniature trucks must be defined. The hardware and software setup of the HIL-simulator should be similar to the one shown in the figure below.

Models

Design kinematic/dynamic models of the miniature vehicles and implement them so that they can be simulated in the LabView PC and design models of the full scale trucks to be simulated in the xPC target machine. The input to the models should be velocity and curvature references, which means that the low level controllers that output the voltages to the propulsion and steering DC-servos of the miniature trucks, and torque requests and inputs to the power steering servos for the full scale trucks, also need to be included in the models.

V2X communication

V2X stands for vehicle-to-vehicle and vehicle-to-infrastructure communication. The position, velocity, acceleration, heading etc. of each vehicle should be available for all others in order for them to coordinate their actions. It should be totally transparent which platform the vehicles exist in: the physical trucks should know where the simulated trucks are, both in LabView and xPC, and vice versa. You are also allowed to let the vehicles and the infrastructure share other types of information; see under Interaction protocol.

Controllers

Several different controllers need to be implemented. First we have the low level controllers that were described earlier: the velocity and curvature controls. The vehicles must also be able to follow the road, i.e. keep their lateral position within the lane. And since we have multiple vehicles on the road we need to have some control over the distances between the vehicles: for example adaptive cruise control that drives at a reference velocity when it can but adapts to the velocity of the vehicle ahead if it catches up on one. And the final step is to coordinate the vehicles in more complex situations such as crossings or ramps, see under Scenarios. The coordination should be implemented in three levels of complexity: 1) Centralized traffic control, for example using traffic lights; 2) De-centralized control based on traffic rules, for example using the right-hand-rule or by giving roads or vehicles pre-defined priorities; 3) cooperative decisions where the vehicles agree on their actions, for example one vehicle slows down to let the other pass the crossing.

Interaction protocol

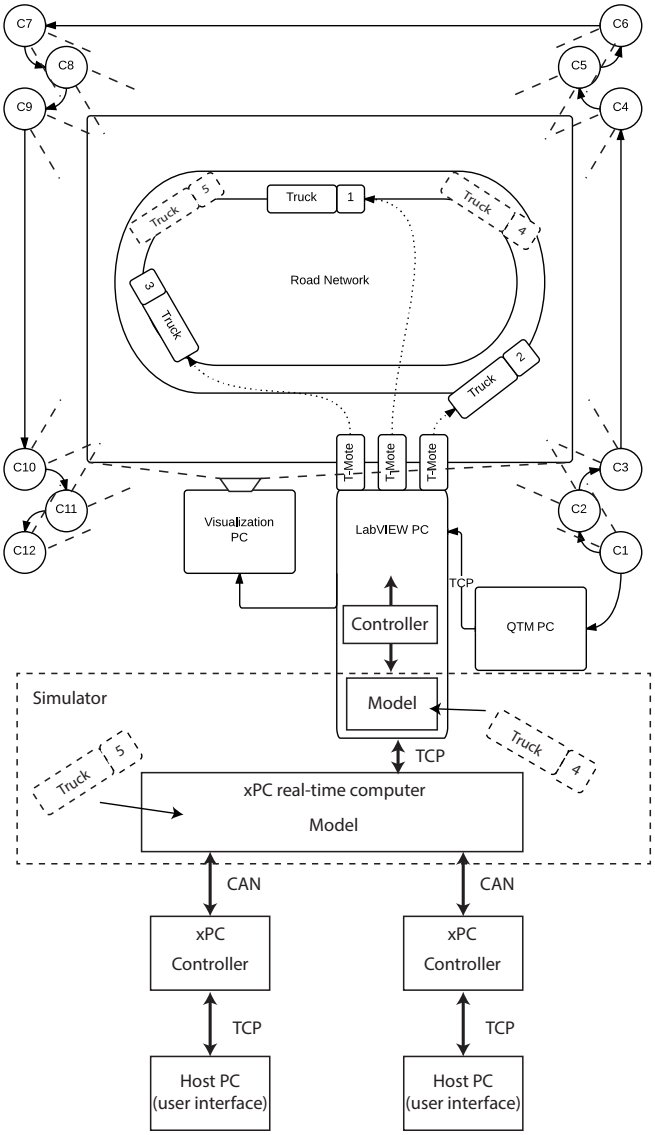
The cooperative control may need some additional information to be shared between the vehicles, such as negotiation, handshaking or status messages. Such an interaction protocol is up to you to define and implement.

Visualization

The simulated vehicles should be represented on the floor as projected images. The wall projector and additional screens can be used to further improve the visualization and the perception of what is happening. When no physical trucks are included, the visualization can be done on a regular screen.

Scenarios

Three traffic scenarios should be implemented and demonstrated: 1) a crossing between two roads; 2) a highway entrance; 3) a complex scenario defined by yourselves.



Figur 3: New integrated HIL simulation platform

Appendix B

Scania GCDC CAN Messages

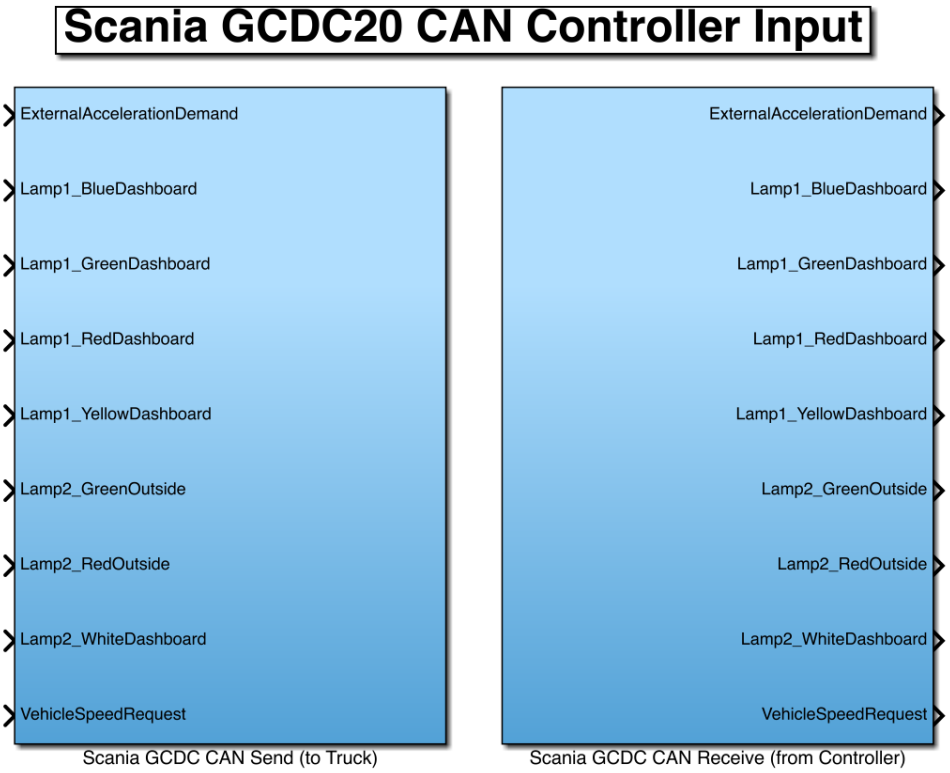


Figure B.1: Simulink Block Sets for GCDC20 CAN Message

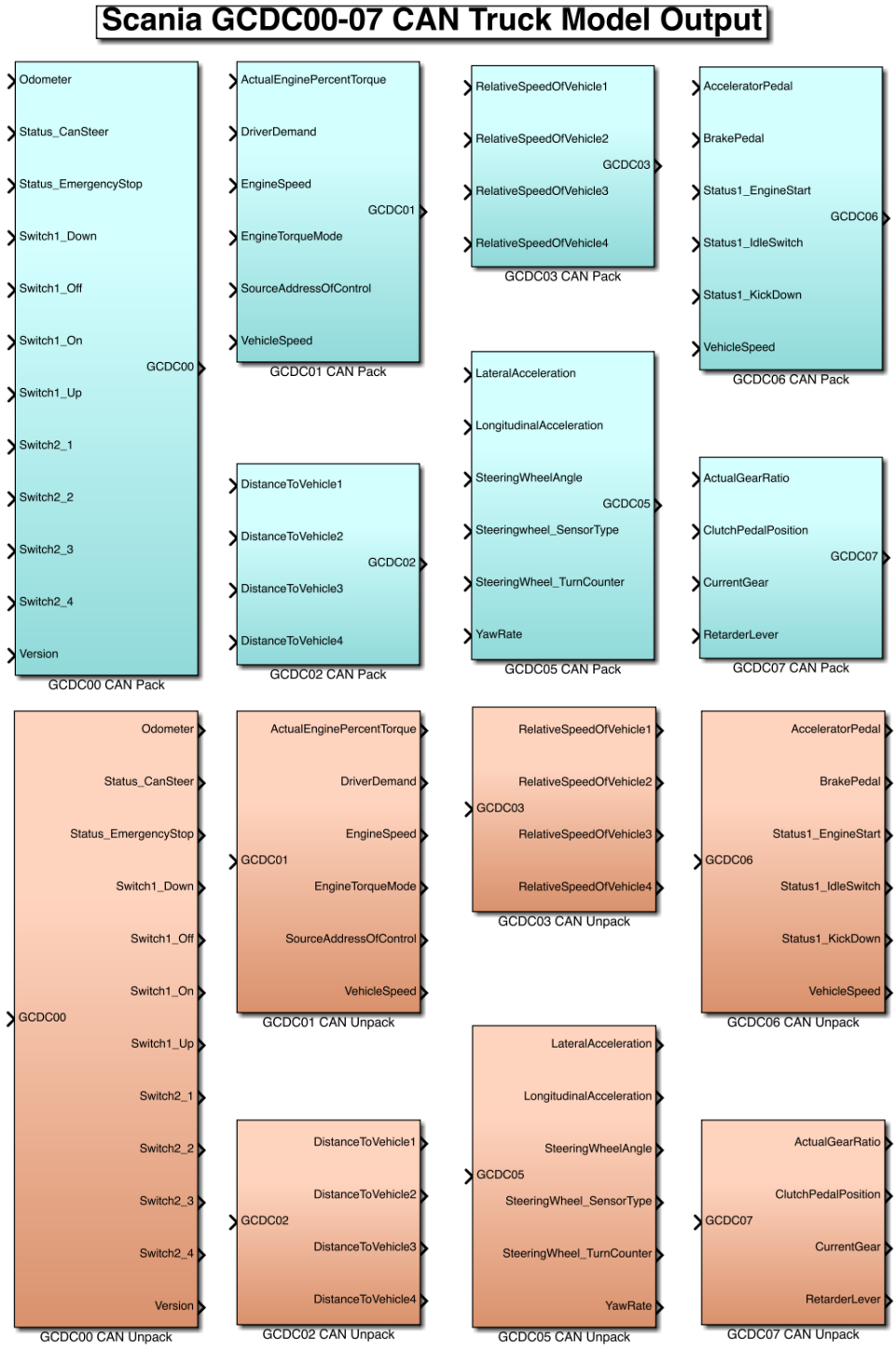


Figure B.2: Simulink Block Sets for GCDC00-07 CAN Messages

Appendix C

Data Packets

C.1 Main Program to Visualization tool

The following is structures of different data packets that are used in the system.

Main Program to Visualization

Double #	Contains
1	Total packet length
2	Number of trucks (physical and virtual)
3	Number of traffic lights
4	Data for traffic signs. What we want to show etc.
5	Spare
6	truck ID
7	truck x
8	truck y
9	truck yaw (degrees)
10	truck Type (0=physical, 1=virtual, 2=xPC, 3=Jonas, 4=disabled)
11	spare
12	Spare
13	truck ID
14	truck x
15	truck y
16	truck yaw (degrees)
17	truck Type (0=physical, 1=virtual, 2=xPC, 3=Jonas, 4=disabled)
18	spare
19	Spare
20	truck ID
21	truck x
22	truck y
23	truck yaw (degrees)
24	truck Type (0=physical, 1=virtual, 2=xPC, 3=Jonas, 4=disabled)
25	spare
26	Spare
	...
5+NoT*7+1	Traffic light 1. Green = 1, Red = 0
5+NoT*7+2	Traffic light 2. Green = 1, Red = 0
5+NoT*7+3	Traffic light 3. Green = 1, Red = 0
5+NoT*7+4	Traffic light 4. Green = 1, Red = 0
	...

(NoT is Number of trucks)

xPC to Main Program

Double #	Contains
1	x position [m]
2	y position [m]
3	yaw [Rad]
4	Velocity [m/s]

Main Program to xPC

Double #	Contains
1	Steering value (0:255)
2	Reference Speed [m/s]

xPC to xPC Monitor

Double #	Contains
1	Velocity [m/s]
2	Engine Speed [RPM]
3	Gear
4	Throttle
5	Brake

Appendix D

Real truck model data

Table D.1: Numerical values and their units for parameters in the real truck model.

Notation	Description	Value	Unit
m	Vehicle mass	12000	kg
A	Frontal area	10.26	m^2
C_d	Drag coefficient	0.6	-
r	Wheel radius	0.495	m
c_r	Rolling resistance coefficient	0.007	-
J_{wheelTot}	Total wheel inertia (4 wheels)	65.8	kg/m^2
L	Length between front and rear axle	4.2	m
$v_{\max}$	Maximum reference velocity	110/3.6	m/s
$v_{\min}$	Velocity below which v_{ref} is set to zero	2	m/s
J_{engine}	Engine inertia	3.5	kg/m^2
n_{idle}	Engine idle speed	500	rpm
T_{friction}	Engine friction	50	Nm
n_{up}	Shift up threshold	1500	rpm
n_{down}	Shift down threshold	850	rpm
η_{final}	Final drive ratio	2.71	-
T_{shift}	Time to shift gear	1	s
J_{mainaxis}	Main axis inertia	0.01	kg/m^2
n_{clutchin}	Engine speed below which clutch pressure is decreased	1000	rpm
$n_{\text{clutchout}}$	Engine speed below which the clutch pressure is set to zero	700	rpm
$\dot{v}_{\text{ret,max}}$	Maximum retardation from brakes	14	m/s^2
$\dot{v}_{\text{ret,th}}$	Hard braking retardation threshold	2.8	m/s^2
$\dot{\phi}_{\max}$	Steering angle rate limitation	± 0.175	rad/s
$\phi_{\max}$	Maximum steering angle	± 0.7	rad
ϕ_{Δ}	Steering angle backlash	0	rad
K_p	Velocity controller proportional coefficient	0.8	-
K_I	Velocity controller integral coefficient	0.2	-
K_b	Velocity controller back-calculation coefficient	1	-

Table D.2: The gear ratios, η_j, used in the model.

j	1	2	3	4	5	6	7
η_j	8.00	5.00	3.02	2.44	1.55	1.24	1.00